

On the Reinforcement Learning Analysis and Learning the Control of Humanoid Robot Leg

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2013

Onder Tutsoy

School of Electrical and Electronic Engineering

Contents

Contents	2
List of Tables	9
List of Figures	9
Abstract:	13
Declaration	14
Copyright Statement	15
Notations	16
Acknowledgements	19
Publications	21
1 Introduction	22
1.1 Overview	22
1.2 Thesis Objectives	23
1.3 Thesis Contributions	24
1.4 Thesis Outline	26
2 Literature Review, Current Approaches and Background	29
2.1 Background and Applications	29
2.1.1 Basic Background for Reinforcement Learning	29
2.1.2 Temporal Difference Learning	31
2.1.3 Various Temporal Difference Learning Methods	32
2.1.4 Reinforcement Learning Applications	34

2.2	Issues in Reinforcement Learning and Proposed Solutions	35
2.2.1	Generalization Issue and Function Approximation	36
2.2.2	Temporal Credit Assignment Problem and TD(λ) Learning	37
2.2.3	Exploration and Exploitation Dilemma	37
2.3	Convergence and Rate of Convergence Analysis	38
2.3.1	Convergence Analysis in the Literature	38
2.3.2	Rate of Convergence Analysis in the Literature	40
2.3.3	Exemplar Test Problems for Reinforcement Learning	41
2.4	Reinforcement Learning Formulation.....	41
2.4.1	Value Function.....	41
2.4.2	Temporal Difference Error.....	43
2.4.3	TD(0) Learning algorithm.....	45
2.4.4	Residual Gradient Learning Algorithm.....	45
2.4.5	Control Policy Learning.....	46
2.5	Current Experimental Humanoid Robots and Control Problems.....	47
2.5.1	ASIMO.....	47
2.5.2	HRP-2.....	47
2.5.3	Wabian	48
2.5.4	iCub.....	48
2.5.5	Control Problems in Humanoid Robotics	49
2.6	Overview of Multi-Body Modelling Software.....	50

2.6.1	Open Architecture Humanoid Robotics Platform (OpenHRP).....	50
2.6.2	Robotran.....	51
2.6.3	MapleSim.....	51
2.7	Summary	52
3	Value Function Convergence Analysis with Bang-Bang Control	53
3.1	Test Problem	53
3.1.1	Exemplar Test Problem.....	53
3.1.2	Test Problem Dynamics	54
3.1.3	Finite Horizon Dynamics	55
3.1.4	Chaotic Terminal Dynamics	56
3.2	Analysis of the Value Function Learning for the Test Problem	57
3.2.1	Finite Horizon Value Function	57
3.2.2	Terminal Value Function	58
3.2.3	Optimal Basis and Parameters for the Value Function	59
3.3	Trajectory Null Space of the TD Error	60
3.3.1	Trajectory Null Space Effect on Residual Gradient Learning	62
3.3.2	Impact of the Trajectory Null Space on the TD(0) Learning.....	63
3.4	Estimating the Higher Order Parameter.....	64
3.5	Value Function Learning with Chaotic Dynamics.....	65
3.6	Balance Test Problem Simulation.....	66
3.6.1	Simulation Details.....	67

3.6.2	Parameter Convergence	68
3.6.3	Rate of Convergence	71
3.6.4	Chaotic Dynamics and Value Function Learning	73
3.7	Summary	75
4	Infinite Horizon Value Function Learning Analysis with Piecewise Linear Control	76
4.1	Test Problem Description.....	76
4.1.1	Exemplar Test Problem and Saturated Piecewise Linear Control	76
4.1.2	Test Problem Dynamics with Saturated Piecewise Linear Control	78
4.1.3	Convergence to the Upright	80
4.1.4	Saturation Boundaries for Convergent Dynamics	81
4.1.5	Value Function.....	82
4.2	Closed Form Terminal Value Function Expression.....	82
4.2.1	Effects of Non-integer Cut off Sample k^*	85
4.2.2	Optimal Polynomial Basis and Parameter Vector.....	85
4.3	Trajectory Null Space of the TD Error	86
4.3.1	Trajectory Null Space Effect on Residual Gradient Learning	89
4.3.2	Impact of the Trajectory Null Space on TD(0) Learning.....	90
4.4	Badly Conditioned Learning Problem	91
4.5	Simulation	92
4.5.1	Simulation Parameters and Algorithm.....	92
4.5.2	Bias and Linear Parameter Convergence	93

4.5.3	Quadratic and Higher Order Parameter Convergence	95
4.5.4	Higher Order Parameter Convergence	96
4.6	Rate of Convergence	97
4.7	Summary	98
5	RL Simulations and Learning Analysis for Higher Order System	100
5.1	RL with Function Approximation and Exploration Noise.....	100
5.1.1	RL with Function Approximation.....	100
5.1.2	RL with Exploration Noise	101
5.2	Simulation Parameters and Setting for RL Analysis	101
5.2.1	Simulation Setting.....	102
5.2.2	Simulation Parameters	102
5.3	RL Analysis with Binary Reward and Bang-Bang Control.....	103
5.3.1	Modified Bang-Bang Control Solution.....	104
5.3.2	Stability Region and Control Surface	105
5.3.3	Learned Bang-Bang Control Policy.....	106
5.3.4	Value Function.....	107
5.3.5	Learned Value Function.....	109
5.3.6	Chaotic and Convergent Dynamics.....	111
5.4	RL Analysis with Quadratic Reward and Bang-Bang Control	112
5.4.1	LQR Based Bang-Bang Control Solution.....	112
5.4.2	Learned Bang-Bang Control Solution.....	113

5.4.3	Value Function.....	114
5.4.4	Learned Value Function.....	116
5.5	RL Analysis with Quadratic Reward and Saturated Piecewise Control	117
5.5.1	Optimal Saturated Piecewise Linear Control Solution	117
5.5.2	Learned Saturated Piecewise Control	118
5.5.3	Optimal Value Function.....	119
5.5.4	Learned Value Function.....	120
5.6	Comparison of the Learning Performance from Failures	122
5.7	Rate of Convergence Analysis.....	123
5.8	Summary	125
6	CPG Based RL Algorithm Learns to Control of a Robot Leg	126
6.1	CPG Controller	126
6.1.1	CPG Control Architecture.....	126
6.1.2	Neural Oscillator Model.....	128
6.1.3	Leg Trajectory Generation	130
6.1.4	Inverse Kinematics Solution	130
6.2	CPG Based RL Modelling in MapleSim and Modelica.....	132
6.3	CPG Based RL Application to Single Robot Leg.....	135
6.3.1	Simulation Parameters	136
6.3.2	Neural Oscillator Output and Learned Leg Trajectories.....	137
6.3.3	IK Solution and Reference Position States	139

6.3.4	Learned Value Function and TD Error	141
6.3.5	Terminal Dynamics of the Leg	142
6.4	Summary	143
7	Conclusion and Future Work	144
	Reference:	147
	Appendix A	154
	Finite Horizon Sample	155
	Closed Form Terminal Value Function Solution with Bang-Bang Control	155
	Appendix B	157
	The Sample at the Saturation Threshold	157
	The Value Function Solution up to Saturation Threshold	158
	The Value Function Solution for the Linear Part of the Control Signal	160
	Appendix C	162
	LQR Feedback Control Strategy	162
	LQR Control Simulation Results	164
	Time Optimal Control Design for an Inverted Pendulum.....	165
	Time Optimal Control Simulation Results.....	167
	Appendix D	169

Word Count: 37.508

List of Tables

Table 1: Pseudo-code of IK solution with Jacobean transposes method.	132
Table 2: Modelica code to re-initialize the robot leg.	134
Table 3: Modelica code for RL feedback signal parameter update.	135

List of Figures

Figure 1: Reinforcement learning components and interaction between them.....	30
Figure 2: RL with three main components: value function, controller and system.	31
Figure 3: Configuration of the TD error, red curve is the value function, dotted blue curve is the temporal difference error and green curve is the reward.....	44
Figure 4: HRP-2 humanoid robot and Asimo robot respectively from left to right, courtesy of Kawada Industries, Inc. and Honda Motor Co.....	48
Figure 5: Wabian HR and child iCub HR from left to right respectively, courtesy of Waseda University and Italian Institute of Technology.....	49
Figure 6 : Dynamics of the system in various regions.	54
Figure 7: The dynamics of the system where the terminal state is zero.	55
Figure 8: a) Chaotic trajectories of the test problem b) tent map based analysis.	56
Figure 9: Configuration of the higher order polynomial learning at the final horizon.	65
Figure 10: The parameter convergence for the bias w_1 and linear w_2 parameters of the residual gradient and TD(0) learning.	69
Figure 11: The parameter convergence of the linear w_2 and higher order parameter w_3 of the residual gradient and TD(0) learning.....	70

Figure 12: RMS parameter error comparison of the residual gradient and TD(0) learning.....	71
Figure 13: Condition number of the regression matrix plotted against discount factor.	73
Figure 14: Value function with digital offset and designed dead zone.....	74
Figure 15: Piecewise linear control, where c is the saturation threshold.....	77
Figure 16: Dynamics of the system in various regions with piecewise linear control, a) convergent dynamics, b) chaotic dynamics around zero.	79
Figure 17: Various dynamical behaviours of the test problem, a) convergent dynamics, b) oscillatory but convergent dynamics, c) chaotic dynamics.....	80
Figure 18: Comparison of the real and closed form value function solutions, a) k^* is non-integer, b) error when k^* is non-integer.....	85
Figure 19: Convergence for the bias w_1 and linear w_2 parameters of the residual gradient and TD(0) learning.....	94
Figure 20: w_3 and w_4 parameter convergence of residual gradient and TD(0) learning.	95
Figure 21: Higher order parameters convergence for the residual gradient and TD(0) learning.....	96
Figure 22: Root Mean Square (RMS) parameter error comparison of the residual gradient and TD(0) learning.....	97
Figure 23: Inverted pendulum model.....	102
Figure 24: State trajectory; a) without noise, b) with noise.	104
Figure 25: Defined stability region.	105
Figure 26: Bang-Bang control decision curve.	106

Figure 27: Learned bang-bang control policy with binary reward.	107
Figure 28: Value function without random control action.....	108
Figure 29: Value function with added random control action.	108
Figure 30: Learned value function with binary reward and bang-bang control.....	109
Figure 31: Value function parameter trajectory.....	110
Figure 32: States of the pendulum after learning, a) with bang-bang control, b) with saturated piecewise control.	111
Figure 33: LQR based bang-bang control solution with state trajectory.	113
Figure 34: Learned bang-bang control solution with LQR reward.....	114
Figure 35: Value function with quadratic reward and bang-bang control.	115
Figure 36: Learned value function with quadratic reward and bang-bang control....	116
Figure 37: Optimal saturated piecewise linear control signal and state trajectory. ...	118
Figure 38: Learned saturated piecewise control.	119
Figure 39: Optimal value function with LQR reward and optimal saturated piecewise linear control.	120
Figure 40: Value function contour graph with LQR reward and saturated piecewise control.	121
Figure 41: The number of failures for each simulation experiment.	122
Figure 42: Comparison of the value functions.....	123
Figure 43: Comparison of the TD errors.....	124
Figure 44: CPG based RL architecture.	127
Figure 45: Configuration of the neural oscillator model.....	128
Figure 46: The neural oscillator model entrainment property [36].....	129

Figure 47: Optimization method based on IK solution.....	131
Figure 48: CPG based RL algorithm model in MapleSim.....	133
Figure 49: Leg model, a) robot leg frame, b) MapleSim 3D visualization.....	136
Figure 50: Learned neural oscillator output.....	138
Figure 51: Learned leg trajectory, a) vertical position, b) horizontal position.	138
Figure 52: The leg model with reduced DOF.	139
Figure 53: Learned reference trajectories, a) for the first joint, b) for the second joint.	140
Figure 54: a) Learned value function, b) TD error.	141
Figure 55: Position trajectory of the leg, a) for the first joint, b) for the second joint.	142
Figure 56: Velocity trajectory of the leg, a) for the first joint, b) for the second joint.	143
Figure 57: Dynamics of the damped inverted pendulum with LQR controller.	164
Figure 58: Control signal of LQR controller for the damped inverted pendulum.	165
Figure 59: Dynamics of the inverted pendulum with continuous time optimal policy.	168
Figure 60: Trajectories of the inverted pendulum with the time optimal control.....	168
Figure 61: Bounds of the chaotic region for the inverted pendulum.	169

Abstract:

Institution: The University of Manchester

Candidate Name: Onder Tutsoy

Degree Title: Doctor of Philosophy

Thesis Title: On the Reinforcement Learning Analysis and Learning the Control of Humanoid Robot Leg

Date: 27 February 2013

Reinforcement learning is a method for learning sequential control actions or decisions using an instantaneous reward signal which implicitly defines a long term value function. It has been proposed to solve complex learning control problems without requiring explicit knowledge of the system's dynamics. Moreover, it has also been used as a model of cognitive learning in humans and applied to systems, such as humanoid robots, to study embodied cognition. However, there are relatively few results which describe the actual performance of such learning algorithms, even on relatively simple problems.

In this thesis, simple test problems are used to investigate issues associated with the value function's representation and parametric convergence. In particular, the terminal convergence problem is analyzed with a known optimal (bang-bang) control policy where aim is to accurately learn the value function. For certain initial conditions, the closed form solution for the value function is calculated and it is shown to have a polynomial form. It is parameterized by terms which are functions of the unknown plant's parameters and the value function's discount factor and their convergence properties are analyzed. It is shown that the temporal difference error introduces a null space associated with the finite horizon basis function during the experiment. This is only non-singular when the experiment is terminated correctly and a number of (equivalent) solutions are described. It is also demonstrated that, in general, the test problem's dynamics are chaotic for random initial states and this causes a digital offset in the value function. Methods for estimating the offset are described and a dead-zone is proposed to switch off learning in the chaotic region.

Another value function estimation test problem is then proposed which uses a saturated piecewise linear control signal. This is a more realistic control scenario and it is also shown to address the chaotic dynamics problem. It is shown that the condition of the learning problem depends on both the saturation threshold and the value function's discount factor and that a badly conditioned learning problem may result. Moreover, it is proved that the temporal difference error introduce a trajectory null space associated with the differenced higher order bases until the saturation threshold of the saturated piecewise linear control signal. These results are then used to explain the behaviour of reinforcement learning algorithms when higher order systems are used and the impact of function approximation algorithms and exploration noise is discussed.

Finally, a central pattern generator based reinforcement learning algorithm is applied to a single leg of a robot where the target is to generate appropriate control signals for each joint.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Statement

- I. The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the “Copyright”) and s/he has given The University of Manchester the right to use such Copyright for any administrative, promotional, educational, and/or teaching purposes.
- II. Copies of this thesis, ether in full or in extracts, may be made only in accordance with the regulations of the John Rylands University Library of Manchester. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.
- III. The ownership of any patents, designs, trade marks and any and all other intellectual property rights except for the Copyright (the “Intellectual Property Rights”) and any reproductions of copyright works, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions can not and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.
- IV. Further information on the conditions under which disclosure, publication and exploitation of this thesis, the Copyright and any Intellectual Property Rights and/or Reproductions described in it may take place is available from the Head of the School of Electrical and Electronic Engineering.

Notations

Abbreviations:

ASIMO:	Advanced Step in Innovative Mobility
CAPI:	Continuous-action Approximate Policy Iteration
CPG:	Central Pattern Generator
Fast-AHC:	Fast Adaptive Heuristic
GSA:	Grow Support Algorithm
GTD:	Gradient Temporal Difference
HR:	Humanoid Robot
HRP:	Humanoid Robot Project
LMS:	Least Mean Squares
LQR:	Linear Quadratic Regulator
LSTD:	Least Squares Temporal Difference
LSTD(λ):	Least Squares combined with TD(λ)
LSPI:	Least Squares Policy Iteration
MPC:	Model Predictive Control
MBE:	Max-Boltzman Exploration
RL:	Reinforcement Learning
RLS:	Recursive Least Squares
RLSTD:	Recursive Least Squares Temporal Difference
RLSTD(λ):	RLS method combined with TD(λ)
RG:	Residual Gradient
RBF:	Radial Basis Function
TD:	Temporal Difference
TD(0):	Temporal Difference with just step back-up
TD(λ):	Temporal Difference with k-step back up
VDBE:	Value Difference Based Exploration
ZMP:	Zero Moment Point

General Symbols, unless stated otherwise:

a_j :	Feedback signal
a, b :	Parameters of the first order system
$A(x_k, w_k^a)$:	Approximated control policy
c :	Saturation threshold for the saturated piecewise control signal
c_g :	Gain of the first order system
c_k :	Centre of the RBF
c_n :	Tonic input
D :	Non-negative diagonal matrix for distribution
e :	Error between current and target positions of the leg
g :	Gravitational acceleration
J :	Jacobian matrix
$J(x_k, u_k)$:	LQR cost function
k^* :	Finite horizon sample
l :	Length of the link
m :	Mass of the link
n_k :	Gaussian noise
$P(x)$:	Position of the end effectors/robot leg
p^t :	Target position
Q :	Positive definite weighting matrix for states
R :	Positive definite/semi definite weighting matrix for control signal
$r(x_k)$:	Immediate reward
$\hat{r}(x_k, w_k)$:	Estimated reward
s :	Activation function
s_k :	Size of the RBF
$s(x_k)$:	$sign(x_k)$
$V(x_k)$:	Exact value function
$\hat{V}(x_k, w_k)$:	Estimated value function
u_k :	Control signal

w_{jk} :	Mutual inhibition of neurons
w_k^a :	Control policy parameter vector
w_k :	Value function parameter vector
x_k :	Discrete position state
x_0 :	Position initial state
$\dot{x}_k$:	Discrete velocity state
$\dot{x}_0$:	Velocity initial state
x_1 :	Position state of the pendulum in modal form
x_2 :	Velocity state of the pendulum in modal form
x_1^f :	Failure position state for the first joint
x_2^f :	Failure position state for the second joint
z_j :	Neuron firing rate
$\dot{z}_j$:	Neuron self inhabitation variable
γ :	Value function discount factor
η :	Value function learning rate
λ :	Eligibility trace
δ_k :	Temporal difference error
$\phi(x_k)$:	Basis function
σ :	Noise standard deviation
σ_0 :	Noise standard deviation constant
η^A :	Control policy parameter update learning rate
τ :	Time constant of the first order system
$\sigma_{\min}^{RG}$:	Minimum eigenvalue of the RG basis
$\sigma_{\min}^{TD(0)}$:	Minimum eigenvalue of the TD(0) basis
γ_n :	Adaptation constant
τ_{CPG} :	Mean membrane time constant
τ_{CPG}' :	Adaptation time constant

Acknowledgements

I would like to thank to my supervisor, Dr. Martin Brown, for his continuous support during my research. His enthusiasm, inspiration, dedication, and rigorous attitude on research work deeply encourage me. His excellent guidance, strong assistance, and critical remarks helped me to make this thesis. I am really glad that I have him as my supervisor. I would also like to express my deep gratitude to my lecturers Kemal Balikci, Ferda Pepesoyler and Cevdet Ademoglu for their unconditional support during every step of my education not just during my PhD.

"Dedicated to my grandfather Mehmet Jutsoy, grandmothers Fmine and Huru Jutsoy and my uncle Harun Jutsoy."

Publications

The key contributions of this PhD thesis are published or submitted to the various journals and conferences.

Journals:

1. M. Brown and O. Tutsoy, "On the value function convergence analysis of temporal difference algorithms with bang-bang control," *IEEE Trans. on Neural Networks and Learning Systems*, submitted.
2. O. Tutsoy and M. Brown "An analysis of value function learning with piecewise linear control," *IEEE Trans. on Neural Networks and Learning Systems*, submitted.
3. O. Tutsoy and M. Brown, "CPG based RL algorithm learns to control of a humanoid robot leg," *IEEE Trans. on Robotics*, submitted.

Conferences:

1. O. Tutsoy, M. Brown, H. Wang, "Convergence analysis of temporal difference learning algorithms based on a test problem," in *The 14th International Conference on Control and Applications (IASTED)*, 2012.
2. O. Tutsoy, M. Brown and H. Wang, "Reinforcement learning algorithm application and multi-body system design by using MapleSim and Modelica," in *International Conference on Advanced Mechatronic Systems (ICAMEchS)*, 2012, pp. 650-655.
3. M. Brown and O. Tutsoy, "An exemplar test problem on parameter convergence analysis of temporal difference algorithms," in *10th World Congress on Intelligent Control and Automation (WCICA)*, 2012, pp. 2925-2930.
4. M. Brown, and O. Tutsoy, "On the analysis of parameter convergence for temporal difference learning of an exemplar balance problem," (Extended abst.) in *Proc. of the 12th An. Con. on Towards Autonomous Robotic Systems*, 2011, pp. 404-405.
5. O. Tutsoy and M. Brown. (2010, Sep.) "Convergence analysis of reinforcement learning approaches to humanoid locomotion," in *UK Automatic Control Conference (UKACC)*, 2010.

1 Introduction

1.1 Overview

There has been interest in using different learning approaches such as Reinforcement Learning (RL) to learn optimal control policy in case of having poor knowledge about system dynamics [1]. The ultimate goal of RL has been to construct efficient, autonomous machines, such as walking robots that can cope with complex and highly dynamic environments. Such an intelligent system should be capable of improving its performance by learning over time in terms of selecting its actions based on gained past experience, so that it can adapt itself to its environment. RL provides learning from consequences of its actions rather than being explicitly taught as in supervised learning. Therefore, the key target of RL is to discover the actions that yield the best reward by performing a trial-and-error process and hence maximize the expected sum of the observed numerical reward called the value function [2]. As the value function is a measure of the quality of the long term learning, understanding key properties of the value function is essential.

Barto and Sutton (1983) developed a RL method based on Temporal Difference (TD) which used the concept of learning from failure [2]. RL was proposed as a method to learn control actions/policies without having to build an explicit model of the system's dynamics [1]. RL was differentiated from other learning approaches in that it uses the evaluated action information, in terms of some form of credit/reward, to learn rather than the given instructions. In 1988, Sutton addressed the temporal credit assignment problem and analyzed some issues associated with parameter convergence [3]. Kenji (2000) derived actor-critic based RL method in continuous time and space and explicitly analyzed exponential eligibility trace formulation, $TD(\lambda)$ [4]. He applied the results to exemplar problems including a nonlinear pendulum and the classic cart pole [5]. Such unstable control tasks are typical of the problem of maintaining balance, using ankle and knee torques, in bipedal robotics. Lewis and Vrabie (2009) formulated the RL with quadratic cost function [6]. They also proposed a value function approximation to solve the TD equations. Xin Xu (2011) proposed a RL method, known as Continuous-action Approximate Policy Iteration (CAPI),

leading fast policy search and adaptive basis function selection in continuous state and action space [7].

Even though RL has been successfully applied to various difficult control/learning problems over the last thirty years, parameter convergence, rate of convergence, which are major properties of the value function learning, have not been extensively investigated even on a lower order simple system. Although it is known that the TD methods have convergence problem; the factors affecting poor learning processes and potential solution techniques are still not fully understood and traditional TD learning is accepted as being an unbiased learning process. However, over the last few years the poor/slow convergence issues has gained more attention by researchers; but they are still unable to clearly express the reason for such behavior, and thus offer an advanced approach to eliminate this problem [4, 8].

1.2 Thesis Objectives

- 1) In this thesis a simple test problem approach is used to analyze the value function learning properties in detail. This leads to a generic approach to reveal key aspects of the TD learning algorithms, particularly TD(0) and residual gradient learning. Considering a test problem allows:
 - To derive closed form value function solution consisting of optimal polynomial basis and correct parameters.
 - To reveal that the character of the closed form value function solution depends on the type of instantaneous reward and control signal.
- 2) This thesis analyzes the value function parameter convergence and rate of convergence for TD(0) and residual gradient learning algorithms in terms of using the closed form value function solution obtained from the simplified test problem approach. This leads:
 - To analyze the value function parameter convergence and rate of convergence in a generic way.
 - To address impact of chaotic dynamics on value function learning and propose a solution to eliminate this effect.

- To express the reasons behind the slow convergence or lack of convergence issue of the TD algorithms.
 - To discuss badly conditioned learning problem of the TD algorithms and propose solutions to avoid this problem.
- 3) Finally in this thesis, reinforcement learning and central pattern generator based reinforcement learning algorithms are applied to an inverted pendulum and a humanoid robot leg with two links. This allows:
- To generalize the insights gained from the test problem value function learning analyses to more general cases when the system is higher order, exploration noise exists and function approximation is performed.
 - To show that central pattern generator based reinforcement learning algorithm applied to the robot leg is able to learn appropriate control signals for each joint.

1.3 Thesis Contributions

- 1) The first contribution of the thesis is to analyze parameter convergence and rate of convergence of value function learning in terms of considering test problems which yield closed form solution consisting of optimal parameters and optimal basis. As there is no such an approach in the literature, this is the first work which clearly analysis the value function learning properties. The convergence analyses are performed on a simplified test problem in chapters 3 and 4, on higher order systems in chapters 5 and 6.
- 2) The second contribution is to proof of existence of a non-zero trajectory null-space associated with higher order parameter until finite horizon sample of value function solution with bang-bang control. Since there is no information available in the higher order basis, the higher order parameter cannot be learned until the finite horizon is reached. This is called finite horizon effect and this causes slow TD learning process. To handle higher order parameter learning at the final sample of the finite horizon, three approaches are proposed in chapter 3 .

- 3) The third contribution is to address impact of chaotic dynamics on value function learning. It is shown in section 3.5 that digital offset in chaotic region of the value function occurs because of the chaotic dynamics. To eliminate this effect a dead-zone control approach is considered in section 3.6.4; hence learning is switched off when the dynamics enter the chaotic region.
- 4) The fourth contribution is to prove existence of non-zero trajectory null space associated with both higher order parameters until saturation threshold of the closed form terminal value function solution with saturated piecewise linear control. Corresponding proof is submitted in section 4.3. It is shown in section 4.5.4 that higher order parameters start learning after the saturation threshold sample and they are functions of the saturation threshold where small saturation threshold yields larger higher order parameters.
- 5) The fifth contribution is to address badly conditioned higher order parameter learning. It is proved in section 4.4 that badly conditioned higher order parameter learning occurs depending on value of the discount factor and saturation threshold. In order to avoid this problem, limits for the discount factor and saturation threshold are specified for the test problem with piecewise linear control.
- 6) The sixth contribution is to analyse rate of convergence of the residual gradient and TD(0) learning algorithms. In terms of using the known correct closed form terminal value function solution parameters, it is shown in sections 3.6.3 and 4.6 that the residual gradient learning algorithm is converging faster than the TD(0) learning algorithm for these test problems.
- 7) The seventh contribution is to analyse the value function and control policy of the RL algorithm for various types of instantaneous rewards and control policies for a higher order system. Optimal solutions are also presented and compared with the learned ones in the chapter 5. It is shown that RL algorithm is able to learn approximated solutions of the optimal ones. The

role of the non-zero trajectory null space and chaotic dynamics on learning results are highlighted.

- 8) The last contribution is to apply CPG based RL algorithm to a single leg of a robot by using MapleSim multi-body modelling software. A symbolic inverse kinematics solution is obtained in terms of utilizing Maple and Modelica advantages. It is shown in chapter 6 that the CPG based RL algorithm is able to learn to generate appropriate control signals to keep the leg at desired position. The CPG and RL learning results are analyzed.

1.4 Thesis Outline

This thesis is organized as follows.

In chapter 2, a literature review of RL is performed. The rational and the history of this approach are briefly described as well as current algorithms which use TD and RG methods for estimating the value function. Current work on the analysis of RL methods is also reviewed, focussing on the convergence and rate of convergence of parametric representations. In addition, existing test problems which have been used to benchmark these algorithms are also described. Despite the fact that RL algorithms have been in existence for approximately 30 years, there is relatively few results available which can be used to give practical insights into the algorithm's efficiency. This is an important gap which this thesis partially addresses, especially in the context of applying the algorithms to legged robots. Existing humanoid robots and multi-body modelling software are also reviewed at the end of the chapter.

In chapter 3, a discrete time unstable, exemplar test problem is introduced. It represents a simplified balance control problem using bang-bang control and a corresponding minimum time or absolute state value function. For certain initial conditions, a closed form expression of the finite horizon, terminal value function can be obtained. The expression is shown to have polynomial/exponential form and its representation in terms of linear and non-linear parameters is described. The parametric basis function expression is used to analyse both the convergence and the rate of convergence of the RG and TD(0) learning algorithms. A null space is shown to exist while the experiment is running, which corresponds to the experiment's

termination, and unless this is appropriately handled, the learning process is biased. It is also shown that, for this experiment, the Residual Gradient (RG) algorithm converges faster than the TD(0) algorithm, a result that contradicts similar comparisons with binary or look-up table basis functions. It is also shown that, in general, this test problem converges to a chaotic attractor and the effect on learning the value function learning is analysed. It is shown that this chaotic behaviour introduces an offset in the previously derived value function expression and a dead-zone is proposed to address this effect.

In chapter 4, a second exemplar test problem is introduced. Again, a simplified balance control problem is used except that now a saturated, piecewise linear control scheme is assumed. The assumption of a linear control scheme implies the use of a quadratic, LQR-type value function. Again, a polynomial-type value function expression is derived and this is used to again perform a convergence analysis of the RG and TD(0) algorithms. The parameter convergence and rate of convergence are again analysed and the null space is again shown to exist during the experiment and the RG algorithm to be again faster than the TD(0) algorithms. This example highlights the explicit dependence on the control variable (unlike in chapter 3 where a bang-bang switching signal is assumed) and reasons for badly conditioned learning problems are described. The poor conditioning is influenced by the limits of the saturation threshold as well as the relationship between the value function discount factor and the system's time constants.

In chapter 5, the RL algorithm is applied to an unstable linear pendulum, which is second order, to analyze both the value function and control policy learning. Two types of instantaneous rewards are considered: binary where the optimal control signal is bang-bang and quadratic where the optimal control signal is piecewise linear. In cases, the learned and the optimal value functions and control policies are compared. The insights gained from the test problem value function learning analysis with function approximation and exploration noise are generalized for a general case where the system is higher order.

In chapter 6, the CPG based RL algorithm is applied to a single leg of a robot and the results are analysed. It is shown that the CPG based RL algorithm is able to learn control actions which satisfy the learning targets.

Finally in chapter 7, conclusions are drawn about the work presented in this thesis future directions are discussed.

2 Literature Review, Current Approaches and Background

RL and TD learning methods have been applied to many complex dynamical systems and some analysis of convergence and rate of convergence has been performed. A key contribution of this thesis is to analyse value function learning for simple systems, in order to obtain insights about the structure of the learning problem and its convergence properties. Therefore this chapter performs a review of the relevant RL literature and key insights about the value function and control policy learning are described. Finally, this chapter reviews existing Humanoid Robots (HR) and multi-body modelling software.

2.1 Background and Applications

This section introduces the two key RL algorithms: TD and RG. Their backgrounds are explained, their structures are analysed and gaps are identified in the convergence analysis reported in the literature

2.1.1 Basic Background for Reinforcement Learning

The history of RL starts in psychology with animal learning which is based on trial-error and reward process [9]. This inspired new initiatives in the artificial intelligence field in the early 1980s [10] and in the late 1980s, links were made with approximate optimal control problems by understanding the role of the value function and dynamic programming. Optimal control problems focus on minimizing a cost function using the Bellman equations [11]. Even though dynamic programming suffers from the curse of dimensionality problem, which causes an exponential growth in the computation burden with the number of states, it has been extensively applied to stochastic optimal control problems. As a result of improvements in these fields, a new approach called TD learning emerged. TD learning is formulated using successive time differences of the reinforcement signal to formulate an error and in this sense it is similar to dynamic programming. All three fields contributed to the creation of a new field called reinforcement learning in the late 1980s [12, 13].

RL is aimed at designing systems with true autonomy and versatility. RL is basically an on-line learning technique which maps an input (situation) to an output (action) using a reinforcement signal (reward) [12, 14]. The instantaneous reward represents an evaluation of the actions which have been taken by the learning agent, as shown in the Figure 1.

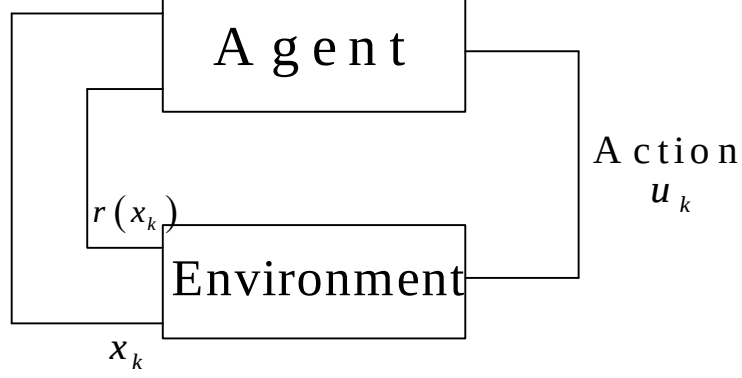


Figure 1: Reinforcement learning components and interaction between them.

In Figure 1, $r(x_k)$ is the immediate reward obtained for each sample point k , x_k represents the state of the system, u_k is the action (also called as policy, actor or controller). Although the reward function provides an immediate assessment of the action, the value function determines the long term performance by summing the total future rewards. For example, the immediate reward of an action could be low but eventually this may yield a higher value function because of future states having larger rewards. The rewards are provided directly, however the value function must be estimated [15]. Therefore, the RL process consists of three main elements: learning prediction (value function) and learning control (control policy) and the system (environment) [16], as shown in Figure 2.

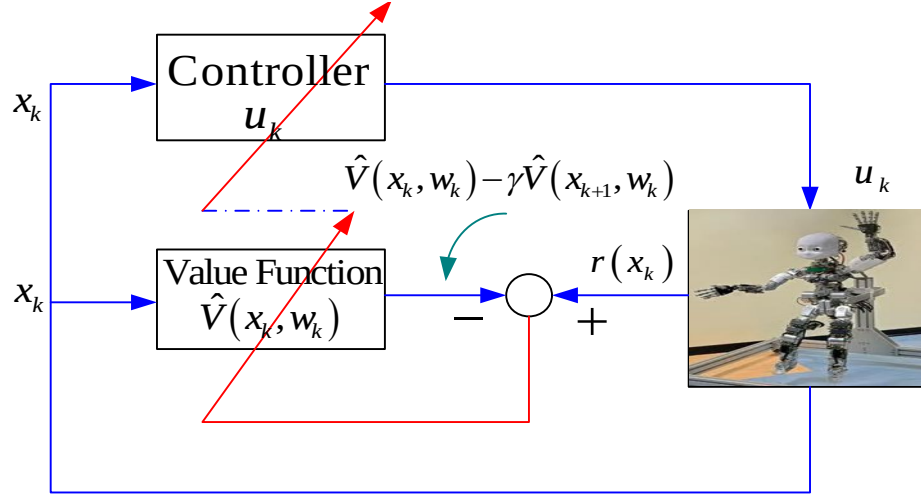


Figure 2: RL with three main components: value function, controller and system.

In Figure 2, $\hat{V}(x_k, w_k)$ is the estimated/approximated value function (also known as the critic) where w_k is the parameter vector [5, 15]. RL has received attention from various fields such as finance, control, and robotic locomotion. Section 2.1.2 introduces the TD learning algorithms from a general perspective.

2.1.2 Temporal Difference Learning

TD learning, which was originally developed by Sutton, is a recursive learning method for approximating the long-term future cost as a function of the current state [2, 17]. The long-term future cost is known as the value function, where the future rewards are discounted exponentially [14, 15]. The TD parameters are updated using successive time differences of the reward signal, [18], which is why it is known as the “temporal difference method”. TD methods learn directly from experience without requiring an explicit model of environment. Moreover, it learns instantaneously before knowing the final outcome of the state transitions and as such requires less memory because of the function approximation capability [19, 20].

The first application of TD type learning appears in Samuel’s (1959) work on playing checkers [21, 22]. The insight behind his learning algorithm was to update the learning agent’s parameters based on an evaluation of the board’s position. The initial information about the board’s position leads to a prediction about the game’s value

when starting from that position. As the positions change during the game, a number of predictions must be made about the possible outcomes of the game. In this learning algorithm, the error is the difference between each pair of successive positions; and from this perspective it is similar to TD learning. Nevertheless, Samuel's learning algorithm has significant differences from the current TD methods [23]. For example, TD methods utilize an evaluated outcome known as instantaneous reward; however Samuel's checker playing algorithm does not use information about the value of its current position. Nevertheless, Samuel's work initiated important research in TD learning in the subsequent years. Section 2.1.3 discusses the development of other learning algorithms based on the initial findings about TD learning.

2.1.3 Various Temporal Difference Learning Methods

TD learning methods are used to estimate or learn the unknown value function. Therefore, the TD algorithm learns to predict the future instantaneous observations or rewards [24]. When the TD method uses a single step back-up for future predictions, then it is known as TD(0) [2, 5]. The discounted value function without any back-up is defined as:

$$V(x_k) = \sum_{t=0}^{\infty} \gamma^t r(x_{k+t}) \quad (1)$$

where $r(x_k) = r(x_k, u_k)$ and γ ($0 < \gamma < 1$) is the discount factor which represents the discount applied to future rewards. The discount factor should be chosen so that the sum in (1) converges. The discounted future reward implies that the current reward is more important than the future rewards in value function learning.

Alternatively, TD(λ) is used to represent the existence of eligibility traces [19, 24-26]. Basically, the eligibility trace is an exponentially discounted technique which is used to assign credit to previously taken actions:

$$e_{k+1} = \lambda \gamma e_k + \frac{\partial \hat{V}(x_k, w_k)}{\partial w_k} \quad (2)$$

$$w_{k+1} = w_k + \eta \delta_k e_k \quad (3)$$

where e_k is the eligibility trace, $\hat{V}(x_k, w_k)$ and w_k are same as described in the section 2.1.1 and η and δ_k are fully defined in the section 2.4.2. Bradke and Barto

developed two new TD algorithms: Least Squares Temporal Difference (LSTD) and Recursive Least Squares Temporal Difference (RLSTD) which are both based on linear, least squares function approximation [27]. They estimate the TD parameters without using an explicit learning rate; hence eliminating a user-defined parameter. The LSTD algorithm estimates the parameters w such that the final value function approximately satisfies the Bellman equation.

$$w = \arg \min_{w \in R^k} \|R + \Phi w - \gamma \Phi' w\|_D^2 \quad (4)$$

$$w = (\Phi^T (\Phi - \gamma \Phi'))^{-1} \Phi^T R \quad (5)$$

where $\Phi = [\phi(x_1) \ \phi(x_2) \ \dots \ \phi(x_{k^*-1})]^T$ and $\Phi' = [\phi(x_2) \ \phi(x_3) \ \dots \ \phi(x_{k^*})]^T$ are matrices consisting of basis functions $\phi(x)$, $R = [r(x_1) \ r(x_2) \ \dots \ r(x_{k^*-1})]^T$ is the reward vector, D is a non-negative diagonal matrix for distribution over the states and k^* is terminal sample point [20]. Even though the LSTD algorithm requires a greater computation effort, it generally achieves more accurate learning in fewer iterations compared to the $TD(\lambda)$ methods. Because the LSTD has an increased computational complexity because of the matrix inversion, its recursive form RLSTD was developed:.

$$w_{k+1} = w_k + P_k \phi(x_k) \left(r(x_k) - (\phi(x_k) - \gamma \phi(x_{k+1}))^T w_k \right) / \left(1 + (\phi(x_k) - \gamma \phi(x_{k+1}))^T P_k \phi(x_k) \right) \quad (6)$$

$$P_{k+1} = P_k - P_k \phi(x_k) (\phi(x_k) - \gamma \phi(x_{k+1}))^T P_k / \left(1 + (\phi(x_k) - \gamma \phi(x_{k+1}))^T P_k \phi(x_k) \right) \quad (7)$$

Bradke and Barto proved the convergence of the LSTD and RLSTD algorithms for periodic and absorbing Markov chains by making a number of assumptions.

Xu analysed the Recursive Least Squares (RLS) algorithms for solving the optimal control problems and stated two TD algorithms based on linear value function approximation [16]. RLS-TD(λ) is a RLS method combined with the TD(λ) which is an extension of the TD(0). They proved convergence and analysed the rate of convergence of the RLS-TD(λ) in terms of ergodic Markov chains. Moreover, it is shown that the RLS-TD(λ) algorithm is computationally more efficient and appropriate for online learning compared to LS-TD(λ). The second algorithm, called the Fast Adaptive Heuristic Critic (Fast-AHC), was developed by utilizing the RLS-

TD(λ) algorithm for estimating the value function. Basically, Fast-AHC applies the RLS method to improve the quality of the value function learning. It is also demonstrated that RLS-TD(λ) has advantages over RLS-TD(0) because λ is closely related to a bound of the approximation error. For example, if $\lambda=0$, then the largest error bound is obtained. On the other hand, $\lambda=1$ yields the smallest error bound.

Motivated by the LSTD algorithm, Lagoudakis and Parr proposed a new approach called the Least Squares Policy Iteration (LSPI) algorithm which combines linear value function approximation with approximate policy iteration [28]. Although, the LSTD is superior compared to traditional TD, as it uses gained past experience effectively, it is not feasible to use it to learn an action without having a model of the process since the LSTD algorithms learns the state based value function of a fixed policy. However, the LSPI is able to learn state-action based value function without requiring a model to select an action. Therefore, the LSPI is a model free and off-policy method. Off-policy TD method implies that the estimated value function is updated not just based on the learned action also the actions which are not considered before [11]. Hence, the off-policy LSPI learns the actions which are not tried during learning. In practice, LSPI was applied to learn the control policy for the swing up pendulum and balancing a bicycle after performing a small number of trials. Section 2.1.4 reviews benchmark applications of RL algorithms in the control and robotics fields.

2.1.4 Reinforcement Learning Applications

While conventional controller design requires the use of a dynamic model, RL does not assume the existence of such a model. RL applications range from robotics to industrial processes and search problems such as computer games [14, 16, 29, 30]. A survey was published by Kaelbling (1996) which focuses on the RL methods and their applications [31]. With respect to balance problems, control engineers have worked on the swing up pendulum task or the problem of balancing a pole on a cart. Kenji (2000) derived the value function and control policy equations based on the Hamilton-Jacobi-Bellman (HJB) equation for infinite horizon, discounted reward problems. He used continuous function approximation and tested the efficiency of continuous time learning problems. Moreover, he also formulated parameter vector update equations

for single step and exponentially weighted eligibility traces [5]. Kimura (1998) expressed the actor/critic equations in discrete time and analysed the role of using eligibility traces of the policy parameters [25]. To derive the actor equations, a Gaussian distribution was used to control the level of exploratory behaviour. They demonstrated the performance of the actor/critic based RL algorithm on a nonlinear cart-pole problem and an inverted pendulum. Gadaleta (2001) introduced a RL based chaos control technique which is simple and flexible [32]. From a control perspective, the task of the controlled chaotic system is to stabilise to an unstable fixed point of the chaotic system. He treated the chaos control problem as an optimal control problem and used the RL as an efficient tool to solve it.

In terms of RL bipedal walking applications, Lin (1991) addressed the question of improving the learning efficiency [33]. He proposed a technique known as experience replay which combines RL and supervised teaching together. Basically, the algorithm is able to learn from previous as well as the current learning examples which speeds up learning considerably and also improves its performance. Hamid (1996) implemented an efficient actor-critic based RL that consists of simple peripheral controllers and small neural networks on a physical robot. Here, for Central Pattern Generator (CPG) CMAC neural network is applied and by using the information coming from each peripheral controller 6 joints of the robot were controlled [34]. Tedrake (2004) investigated and applied a stochastic RL algorithm to a 3D robot [35]. He used a simplified version of a sophisticated HR which has only four actuators controlled in frontal and sagittal planes. Thus, the robot mechanical design is based on passive dynamic walker. Once the robot learns the optimal control policy, it is able to adapt itself to small changes. Gen, Morimoto, Matsubara, Nakanishi and Cheng (2008) successively implemented a continuous time RL algorithm on a full body 3D HR [36]. They applied CPG-based optimal control with policy gradient method to the 3D robot. Section 2.2 discusses the issues in RL.

2.2 Issues in Reinforcement Learning and Proposed Solutions

Since the development of TD learning in the 1980s, problems associated with TD learning such as generalisation, temporal credit assignment, and parameter convergence have received attention. In this part of the thesis, a literature review of these issues and proposed solutions is performed.

2.2.1 Generalization Issue and Function Approximation

When RL methods are applied to tasks, they will encounter many states which have not been experienced before. This situation almost always exists when look-up tables are used to represent the value function and/or control policy. In Sutton's TD learning algorithms look up tables, which are discrete representation of the value function, are extensively used. However, look up tables suffer from the curse of dimensionality where the number of parameters is exponentially dependent on the number of states [23, 37]. An alternative is to use continuous function approximation approaches which, in general, require considerably fewer parameters. With regard to a look up table, in order to learn the function, every discrete state in the state space must be visited. Moreover, as Sutton stated in his convergence theorem, it is required that the states should be visited infinitely many times to provide guaranteed convergence [17]. This is not achievable in the real world due to insufficient storage capacity and also time. Furthermore, Boyan and Moore (1995) demonstrated that function approximation with look up tables in dynamic programming is not robust and may yield divergence from the optimal solution [38]. They proved the results by performing a simulation and proposed a new robust stable generalising function approximation called Grow Support Algorithm (GSA). This algorithm is based on the approximation which represents the final optimal value function without relying upon intermediate values.

Using continuous, low dimensional models should facilitate learning complex tasks in high dimensional space as they can generalise knowledge to areas of the state space which have not been experienced. However, in the case of learning complex goals, a limited number of non-linear basis function may cause poor representation. In addition, Tsitsiklis (1996) stated that the value iteration with linear approximation may yield divergence in parameter convergence [39]. Moreover, gradient descent with structural credit might converge to local optima rather than global one [23]. This is called the generalization problem which is also known as the structural credit assignment problem. Usually, the input space is too large for exploration; hence the learning algorithm must have capability to generalise knowledge between similar situations and actions in terms of using the experience gained from similar situations. The most common way of dealing with this problem is using artificial neural

networks for approximation, since the parameterised representation has a relatively small number of parameters to be estimated [2, 40, 41]. However, nonparametric representations such as kernel regression and support vector machines can also be considered [11]. All these methods are heuristic/approximation techniques; hence this leads conservative convergence proofs and significant modelling error may yield poor or even divergent results. Section 2.2.2 discusses the temporal credit assignment problem.

2.2.2 Temporal Credit Assignment Problem and TD(λ) Learning

Determining the contribution to an experiment's success or failure of a particular action from a sequences of actions is known as temporal credit assignment problem [23, 31, 18]. This is obviously important in RL, however similar problem in adaptive control exists, for instance when the model's order is wrong, then this can cause learning instability. The question issue with temporal credit assignment is how an action taken at particular step affects the terminal outcome and how the learning algorithm adapts the parameters in order to improve the overall performance. The temporal credit assignment problem resulting from delayed reinforcement cannot be solved completely. When the delayed reinforcement is known in advance or the reinforcement is always delayed by a fixed amount, then the created assignment problem can be treated easily. When the time delay in the reinforcement signal is not accurately known, the learning problem is difficult. Sutton (1984) stated that temporal credit can be assigned based on frequency and recency of successful past applications [42]. This approach is known as TD(λ) learning which uses eligibility traces to assign credit for the desired actions. Tsitsiklis and Roy (1997) pointed out that the TD(λ) convergence is not guaranteed with nonlinear value function approximation [19]. However, they showed that TD(λ) with linear function approximation is convergent. A further problem in TD learning is the exploration and exploitation dilemma which is discussed next.

2.2.3 Exploration and Exploitation Dilemma

In RL, exploration is necessary in order to determine both the optimal value function and the optimal policy. Balancing the amount of exploration and exploitation is a challenge in RL since this directly affects the quality of the learned control policies

and also the rate of convergence. If purely exploratory behavior is used, then the random exploratory actions may yield negative (failure) instantaneous rewards. On the other hand, just using exploitative actions may prevent the learning algorithm from maximizing the value function [43, 44]. Therefore, there must be a balance between the amount of exploration and exploitation based on the learning problem and also the learning parameters such as the learning rate and discount factor.

ε – Greedy policy selects an equally distributed random action having probability $0 \leq \varepsilon \leq 1$ to trade-off exploration and exploitation [2]. Equally-distributed action selection might lead to poor instantaneous rewards as a result of poor action selections. A Boltzmann distribution (Softmax policy), which takes into account the estimated value function, has been proposed to overcome this problem [2, 45]. This weakness can be eliminated by combining the Softmax and ε – Greedy exploration approaches. Max-Boltzman Exploration (MBE) selects exploration actions based on Softmax policy rather than equally distributed action selection [44]. Choosing appropriate exploration parameters for all of these approaches is time consuming and bad choices yield poor learning performance. Value Difference Based Exploration (VDBE) Softmax adapts the exploration rate based on changes in the value function [43]. The next section discusses convergence analysis of TD learning.

2.3 Convergence and Rate of Convergence Analysis

This part of the thesis presents a literature review for parameter convergence and rate of convergence analysis of the TD algorithms.

2.3.1 Convergence Analysis in the Literature

Even though TD learning is a simple and elegant learning method, a comprehensive and rigorous analysis of the TD learning algorithms and their parameter convergence has not been established. Sutton analysed the convergence of TD(0) and showed that the update in the value function tends to zero as the number of samples goes to infinity [2, 17, 46]. He also analysed the role of learning rate selection to ensure convergence. While these results are significant, they are only valid for systems where the number of parameters is equivalent to the number of the states (look-up tables). Dayan analysed the effect of using value function approximation [12, 47]. He

analysed parameter convergence when the function approximator consists of a set of basis functions and tuneable weight parameters. However, this form of convergence proof is not strong; it does not take into account the approximation error bounds. Schapire and Warmuth proposed a worst-case analysis approach for TD learning which is based on Sutton's algorithm [48]. The key difference between these results is that Sutton makes statistical assumptions such as stochastic nature of the environment but Schapire and Warmuth consider no statistical assumptions. Nevertheless, their analysis does not fully address parameter convergence, rate of convergence and error bounds.

Tsitsiklis and Van Roy presented a convergence analysis for TD learning with linear function approximation [19]. They introduced TD learning convergence limits and bounds for the approximation error. They showed that a stochastic approximation algorithm in general converges if the added random noise is large enough. This work also emphasized the role of online parameter update on optimal parameter convergence. However, even though this work provides better insights into the performance of the TD learning algorithms, they only focused on Markov chain which is geometrically ergodic and the state space of the chain is countable. Tadic generalized and extended the convergence results presented by Tsitsiklis and Van Roy with fewer but broader assumptions [49]. He analyzed the asymptotic convergence properties of the TD learning algorithms with linear function approximation. He carried out his research for a discounted, uncountable Markov chain with an uncountable final dimensional state space problem. He showed almost sure convergence of the TD learning algorithms and established an upper bound for asymptotic approximation error.

Sutton, Szepesvari and Maei introduced the first off-policy TD algorithm which is stable in the case of linear function approximation for a Markov chain where complexity of the algorithm increases with the number of the parameters [50]. They proposed the gradient temporal difference (GTD) algorithm and proved that under stochastic approximation assumptions this algorithm converges to the same least squares temporal difference (LSTD) solution. Nevertheless, empirical results revealed that GTD(0) is not fully capable of efficiently solving the off-policy and also the on-policy learning problems. Maei introduced the first TD algorithm which is based on

Sutton's work for nonlinear function approximation [8]. He proved almost sure asymptotic convergence of two TD algorithms for any finite Markov decision process to a locally optimal solution. TD algorithms are incremental and the computational complexity linearly increases with the number of parameters. To overcome the instability issue, a weight update projection technique is used. Promising empirical results demonstrate the efficiency of the algorithms. However, these empirical results just compared the proposed algorithms with TD(0) in terms of the learned value function and mean square error for the estimated value function. They do not provide clear insights about the parameter convergence and rate of convergence as the correct parameters and the optimal basis for the TD algorithm are not specified.

2.3.2 Rate of Convergence Analysis in the Literature

The first theoretical proof which compares the speed of convergence of TD(0) and Residual Gradient (RG) learning was presented by Schoknecht and Merke [51]. They considered the asymptotic rate of convergence which is based on the relationship between the initial and final error. The asymptotic convergence rate depends on learning rates selected with $\eta^* = 2/(|\lambda_1| + |\lambda_F|)$, where λ_1 and λ_F are maximum and minimum eigenvalues of TD(0) correlation matrix. It is claimed that the TD(0) converges asymptotically faster compared to the residual gradient learning with a value function having binary basis function (look-up table). However, it is also mentioned that the residual gradient learning might converge faster than TD(0) when using more general value function approximation schemes. A worst case error comparison analysis of these two learning algorithms was carried out when a linear function approximation is used [52]. The learning rates are defined in terms of the discount factor γ and the basis. It was concluded that the residual gradient learning yields smaller TD error, but they do not analyse the rate of convergence.

Bradke and Barto analysed the rate of convergence of the TD algorithms in a different way [27]. To measure the efficiency of the algorithms, they defined the TD error variance of a Markov chain and showed that the rate of convergence of the TD algorithms linearly depends on this TD error variance. By using the TD error variance approach, they claimed that the developed TD algorithms converge faster than the existing TD algorithms. However, if the TD learning algorithm is biased, then

comparing the TD error variance will not give any insights about the rate of convergence. Thus, it is essential to explicitly analyse the parameter convergence and rate of convergence properties of the value function for better understanding.

2.3.3 Exemplar Test Problems for Reinforcement Learning

Using test problems to numerically evaluate the performance of a TD algorithm can reveal important information about the learning method's properties. However, few/no exemplar test problems exist in the literature where closed form expressions of the value function and control policy can be used to benchmark different learning algorithms. A simple benchmark time optimal control problem was used by Dier and Joshi to compare the corresponding bang-bang solution with the RL solution [53]. They were able to show that the RL algorithm is able to generate a bang-bang control policy. However, they did not analyze the parameter convergence and rate of convergence of the value function for this test problem. The next chapter introduces a simple, but representative, test problem which allows analysis of the convergence and the rate of convergence for the terminal value function.

2.4 Reinforcement Learning Formulation

To analyse the parameter convergence of the learned/approximated value function and to learn the optimal control policy, the relevant RL update equations are derived in either continuous or discrete time. In this part of the thesis, the relevant RL equations will be derived.

2.4.1 Value Function

TD is a recursive learning method for estimating the value function, $V(x_k)$, defined in equation (1) [5, 15]. In this expression, the dependence on the control action, u_k , has been suppressed. In practice, experiments are performed for a finite time, so the upper limit in (1) is often k^* , which generally depends on the initial state [2, 17].

As with all optimal control criteria, the form of the trajectory and the end-point costs determine both the system properties and the form of the control scheme. Choosing a binary reward (success/failure), for example, produces minimum time behaviour while a quadratic, state-dependent reward generates Linear Quadratic Regulator

(LQR) type optimal responses [54]. An implicit assumption in these results is the ability of the RL algorithm to efficiently estimate the value function, which is especially important when the system's dimension and complexity increase [14].

Approximating the value function is usually done using flexible parametric or nonparametric methods. Parametric approximation techniques map a parameter space into the value function or policy space. The number of parameters and the basis functions are usually determined in advance. For example, a linear in the parameters model is given by:

$$\hat{V}(x_k, w_k) = \phi(x_k)^T w_k \quad (8)$$

where $\hat{V}(x_k, w_k)$ is the approximated value function and w_k is the parameter vector. Chapter 5 of this thesis uses this kind of function approximation while chapters 3 and 4 use an optimal basis for the terminal value function. Linear parametric approximation schemes such as state aggregation, tile coding and normalized Gaussian radial basis function (RBF), are widely used in the RL literature as the theoretical analysis is simplified and the rate of parameter convergence is often faster [14, 38, 55]. Tile coding is simple and computationally efficient and even though tile coding is a discrete representation of the (continuous) state space, its generalization capacity is reported to be preferable to straight discretization/look-up tables. Nonetheless, it is often necessary to determine the size of each tile intuitively [56]. RBFs are local basis functions which can be configured to have good approximation capabilities when the topology (number of hidden units, position, widths etc.) is appropriately set. However, if there is no accurate prior knowledge to define this number, then non-linear, on-line schemes can be used to improve the representation or regularization methods can be used to try and automatically control the complexity of the network [55]. The relationship between fuzzy and RBF schemes is well known, and most of the previous comments also apply to fuzzy approximation schemes [57].

Kernel based approximation schemes are termed nonparametric as the model's complexity is a function of the training data. Typically, such schemes use regularization/smoothing techniques to control complexity. Nonparametric approximation methods are able to adapt the model's complexity to the data by reducing the approximation bias and this can be an advantage when the data is scarce

but also a disadvantage if the data is plentiful. This is particularly challenging for online (RL) methods as new data is continually being gathered [55].

2.4.2 Temporal Difference Error

From equation (1), the value function is defined as the sum of the discounted future rewards [2]. Therefore, the value function can be written as:

$$V(x_k) = r(x_k) + \gamma r(x_{k+1}) + \dots + \gamma^\infty r(x_\infty) \quad (9)$$

Equation (9) can be expressed in terms of the current reward and future value function, which motivates the “differenced error” used in the TD learning.

$$V(x_k) = r(x_k) + \gamma V(x_{k+1}) \quad (10)$$

This must hold for any optimal/non-optimal policy [6, 19]. This defines an optimality condition which is satisfied when the estimated value function is accurate. Otherwise, a temporary error occurs between the instantaneous reward and the differenced value functions. This defines the TD error between the instantaneous reward, $r(x_k)$, and estimated reward, $\hat{r}(x_k, w_k) = \hat{V}(x_k, w_k) - \gamma \hat{V}(x_{k+1}, w_k)$, [5]. Figure 3 illustrates the instantaneous and expected rewards and the corresponding TD error.

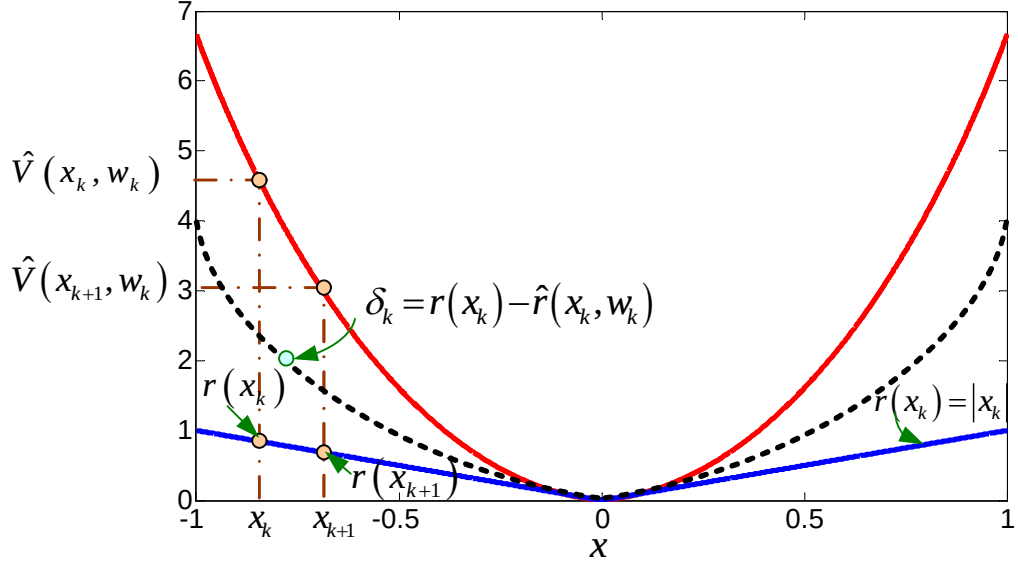


Figure 3: Configuration of the TD error, red curve is the value function, dotted blue curve is the temporal difference error and green curve is the reward.

In Figure 3, the reward is the absolute value of the state, $x \in [-1, 1]$ and the value function discount factor $\gamma = 0.85$. The instantaneous error between the true and the estimated reward is given by:

$$\delta_k = r(x_k) - \hat{r}(x_k, w_k) \quad (11)$$

where δ_k is known as the TD error [5]. Expanding the estimated reward in (11), using the approximated value function representation (8), gives:

$$\delta_k = r(x_k) - (\phi(x_k) - \gamma\phi(x_{k+1}))^T w_k \quad (12)$$

The TD error is equivalent to the error used in many instantaneous learning algorithms in the literature, such as least mean square (LMS) algorithm. However, the model's output and the corresponding basis functions is differenced, hence it should not be surprising that the learning rate should be scaled to compensate for the differenced basis function and also information may be due to the target reward representing the differenced target value function. This is further discussed later in the thesis.

2.4.3 TD(0) Learning algorithm

A family of TD learning methods has been developed which attempt to speed up convergence by filtering the previously visited state vectors. When there is no filtering, it is known as TD(0) learning [2, 5] and the parameter update rule is:

$$w_{k+1} = w_k + \eta \delta_k \frac{d\hat{V}(x_k, w_k)}{dw_k} \quad (13)$$

where η is the learning rate. The TD error is defined with respect to the reward error but the model sensitivity is a function of the current value function. For linear in the parameters models, this becomes:

$$w_{k+1} = w_k + \eta \delta_k \phi(x_k) \quad (14)$$

where the learning rate lies between $0 < \eta < 2 / \left(\left(\phi(x_k) - \gamma \phi(x_{k+1}) \right)^T \phi(x_k) \right)$ for stable TD(0) learning where it is assumed $\left(\phi(x_k) - \gamma \phi(x_{k+1}) \right)^T \phi(x_k) > 0$. Although the TD(0) algorithm with binary basis functions (look up table) is guaranteed to converge to the optimal solution, this is not the case for more general basis functions and the residual gradient learning algorithm was developed to overcome this problem [51, 58].

2.4.4 Residual Gradient Learning Algorithm

The residual gradient learning algorithm was investigated by Baird to tackle the divergence problem associated with TD(0) learning [58]. It guarantees convergence to a local optimum when gradient descent is performed on the instantaneous squared TD error:

$$E_k = \frac{1}{2} |\delta_k|^2 = \frac{1}{2} \left| r(x_k) - \left(\hat{V}(x_k, w_k) - \gamma \hat{V}(x_{k+1}, w_k) \right) \right|^2 \quad (15)$$

The residual gradient algorithm performs instantaneous gradient descent on E_k .

$$\begin{aligned} w_{k+1} &= w_k - \eta \frac{dE_k}{dw_k} \\ &= w_k + \eta \delta_k \frac{d\hat{r}_k}{dw_k} \end{aligned} \quad (16)$$

Here, the model sensitivity vector is with respect to the estimated residual rather than the value function as in the case of TD(0) and for linear in the parameters models.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta \delta_k [\boldsymbol{\phi}(x_k) - \gamma \boldsymbol{\phi}(x_{k+1})] \quad (17)$$

This is simply the usual LMS estimation algorithm, where the error is the reward error and the model sensitivity vector is the weighted difference between successive basis function vectors as the error is in the residual rather than the value function space. Learning is stable (a prior error is reduced) when the learning rate satisfies $0 < \eta < 2 / \|\boldsymbol{\phi}(x_k) - \gamma \boldsymbol{\phi}(x_{k+1})\|_2^2$. The usual techniques should also be used to counter systematic modelling error and noise.

2.4.5 Control Policy Learning

It is also necessary to learn the optimal control policy together with the value function. This is an actor-critic based RL where critic is the value function and the actor is the control policy as described in the section 2.1.1. The discrete time value function equations were introduced in the section 2.4.3 and 2.4.4. Here, the control policy learning equations are formulated. Consider the policy implemented by the actor as:

$$u_k = s\left(A(x_k, \mathbf{w}_k^a) + \sigma n_k\right) \quad (18)$$

where n_k is the noise used for the exploration purpose, s is a sign function or monotonically increasing output function, σ is the standard deviation for the noise described as $\sigma = \sigma_0 \min\left(1, \max\left(0, (V_{\max} - \hat{V}(x_k, \mathbf{w}_k))\right)\right)$. Here $V_{\max}$ is the maximum value of the estimated value function and σ_0 is a constant. $A(x_k, \mathbf{w}_k^a)$ is a function approximator with basis function and controller/policy parameter vector:

$$A(x_k, \mathbf{w}_k^a) = \boldsymbol{\phi}(x_k)^T \mathbf{w}_k^a \quad (19)$$

where $\boldsymbol{\phi}(x_k)$ is the basis function same as the one defined in the section 2.4.1 and $\mathbf{w}_k^a$ is the actor parameter vector to be learnt [25, 59]. In order to update the parameters so that the actor reaches the optimal control policy, the TD error is used as a reinforcement signal and the same noise signal is added to ensure that the parameter update sign is in the correct direction. The policy weight update rule is given by:

$$w_{k+1}^a = w_k^a + \eta^A \delta_k n_k \frac{\partial A(x_k; w_k^a)}{\partial w_k^a} \quad (20)$$

where η^A is the learning rate for the actor policy update [5, 25].

2.5 Current Experimental Humanoid Robots and Control Problems

While this thesis is primarily aimed at RL methods and their analysis, the work was motivated by simple balance problems in robotics, in particular bipedal locomotion. This is a complex but topical area as a number of successful bipedal robots have now been developed which demonstrate walking, running, and learning/cognitive functions. However, the challenge is to improve performance of the robots in order to mimic humans. This section briefly reviews four current humanoid robots.

2.5.1 ASIMO

Advanced Step in Innovative Mobility (ASIMO), shown in Figure 4, is a HR designed by Honda in Japan. It is 130 cm tall and 54 kg weight. ASIMO is able to walk and run with a speed up to 6 km/h. A camera is mounted on its head and is used for both visualization of the surface and also measuring the distance. It can follow moving people and greet people coming towards it. Moreover, Asimo is capable of recognising the posture and gestures of humans. In addition, it can be commanded by voice and human gestures. It is able to distinguish different noise/voice sources and interpret the face mimics of people. On the other hand, there are three main disadvantages to overcome; ground impact forces must be properly absorbed, flexible movement of the upper body/legs must be provided and slipping/spinning problem must be addressed. In addition, Asimo HR is not durable enough and battery power lasts less than an hour [60].

2.5.2 HRP-2

Humanoid Robot Project 2 (HRP-2), shown in the Figure 4, has been developed by Japanese manufacturing science and technology centre for domestic tasks. Its height is 154 cm and weight is 58 kg. Its feet enable it to walk on uneven terrain. It has 30 degrees of freedom and is able to walk in narrow places using a flexible hip and joint structure. It can keep its balance on one foot and it is able to talk but not fluently. In

addition, it can lie down and stand up slowly; it is able to see due to three cameras mounted on its head but it cannot hear.

Given locomotion parameters such as step length, step height and limit cycle duration, the Zero Moment Point (ZMP) approach is able to generate a smooth walking gait. HRP-2 is able to take larger, faster steps due to generated ZMP trajectories. However, the generated trajectories are unable to consider any type of collision and environmental effects [61].



Figure 4: HRP-2 humanoid robot and Asimo robot respectively from left to right, courtesy of Kawada Industries, Inc. and Honda Motor Co.

2.5.3 Wabian

WAseda BIpedal humANoid (WABIAN) shown in Figure 5 was designed by the University of Waseda in Japan. Its height is 1.5 m and weight is 64.5 kg; the robot has 41 DOF and is able to mimic human walking in terms of using heel-to-toe strike. Moreover, the toe joints are passive; hence there is no need for actuator and controller design, which makes it lighter. It walks on uneven terrain and when it falls down it uses control techniques together with a newly developed foot mechanism to stand up. Moreover, it can carry an external battery and is able to switch between batteries. On the other hand, it has little emotional expression patterns and limited ability to interact with humans/environment [62]. The robot should be equipped with essential intelligent capabilities such as learning from failures.

2.5.4 iCub

The iCub, shown in Figure 5, was developed to mimic a toddler HR and was funded by the European Commission. It was developed as a cognitive test bed, where

researchers could evaluate how learning and human-like reasoning processes would perform when an artificial entity interacted with the real environment. For example, researchers have tried to teach iCub to talk like a child learning from parents. It attempts to model the learning processes of a toddler. Moreover, it will be able to learn from experience and adapt itself to environmental changes when the project is completed. Its height is 105 cm; weight is 20.3 kg and has 53 DOF.

Although the iCub robot has been developed to evaluate cognition research, it does not have full locomotion capability to perform advanced tasks such as walking at variable speeds and in unstructured terrains [63].

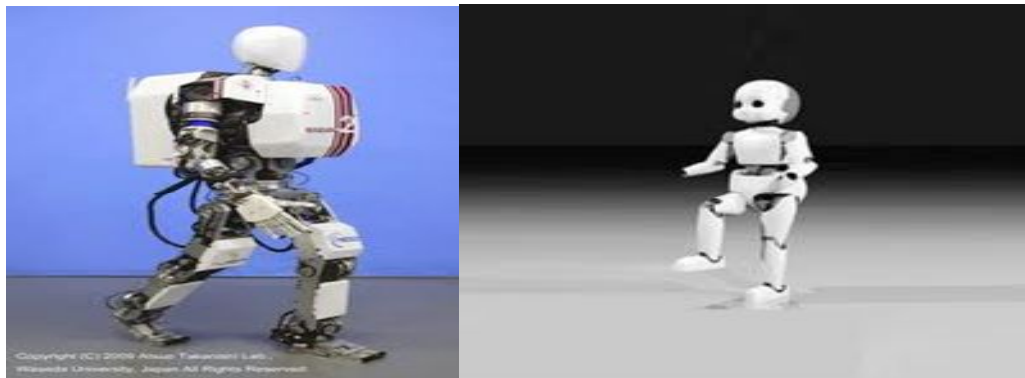


Figure 5: Wabian HR and child iCub HR from left to right respectively, courtesy of Waseda University and Italian Institute of Technology.

2.5.5 Control Problems in Humanoid Robotics

Although considerable progress has been made in the mechatronic design of such HR and advanced control techniques, unsolved issues such as energy efficiency, complete autonomy, and user friendly safe HRs need to be addressed. In addition, advanced locomotion in unstructured and complex domains is still a problem for many HRs as they are unable to cope with large movements, variable speeds and various constraints [64]. As a result, current HRs still suffer from a lack of robustness, versatility, and adaptability that biological systems possess.

In order to address the stability and periodic walking issues, Centre of Pressure (COP) and Zero Moment Point (ZMP) concepts are widely implemented. However, a biped robot needs to act like a human, so that it must be capable of learning new gaits in the case of moving in unknown terrains. Thus, these force control techniques should be

combined with learning strategies, because learning allows the HR to modify its dynamic walking pattern which is necessary for autonomous walking. Moreover, HR control techniques rely on kinematic and dynamic modelling of the mechanism which requires complete state measurement and interaction between the feet and ground. Consequently, these approaches result in heavy computation and optimization [65].

2.6 Overview of Multi-Body Modelling Software

Obtaining realistic mathematical models of a physical multi-body system, such as walking robot, is a laborious but necessary process for multi-body system design and analysis. The generated mathematical models are also a key part of the controller design process. Computational tools are necessary to derive such a mathematical model, and a number of commercial and academic multi-body simulation packages have been developed. In this part of the thesis, existing multi-body modelling software tools are reviewed and the pros and cons of each are highlighted.

2.6.1 Open Architecture Humanoid Robotics Platform (OpenHRP)

OpenHRP (Open Architecture Human-centred Robotics Platform) was developed to initiate advanced humanoid robotics research by providing an open robotic software and hardware platform. It contains a number of software components and libraries that are essential for robotic applications. A library consisting of position, torque, vision, and inclination sensors are important for designing controllers for robots. Walking pattern generators can produce stable bipedal walking trajectories for the designed models. Moreover, this software includes useful tools for the walking and balancing such as collision detection. A 3D visualization interface allows users to see the model and its behaviour during the simulation. Its capability to generate compatible simulation experiments with physical real world application is shown on the HRP robots [66, 67]. Although this software is free and open software, it can be directly only used for HRP robots. It is not feasible to apply it to other robots as it does not allow the user to modify or customize major features of the OpenHRP. In addition, the available documentation is limited and technical support is not provided [68].

2.6.2 Robotran

Robotran is a symbolic multi-body modelling and simulation software package developed at the Universite Catholique de Louvain in Belgium. It generates symbolic equations of motion whose validity has been confirmed on the CoMan HR [69]. It has three major features: graphical user interface, symbolic equation generator and interface between Robotran and Matlab. Documentation on modelling and technical help on software usage is poor and online help is insufficient. It cannot generate C-code of the model directly and it does not provide visualization of walking [68]. MapleSim, a newly launched symbolic modelling tool for creating and simulating complex physical systems, is able to generate stand alone C-code for real time implementation. This allows controllers and models to be accurately exported to real systems [69].

2.6.3 MapleSim

MapleSim is a multi-body and multi-domain (electrical, hydraulic, and pneumatic physics) modelling tool. It is possible for users to obtain models by using multi-body components from various engineering fields. A case study for modelling the CoMan HR with the MapleSim has been performed to examine bipedal locomotion [70]. MapleSim has a rich component library covering more than 400 components and can be used to design electrical, hydraulic, 1D and multi-body mechanical models where most of the components are from the Modelica standard library 3.1. In addition to the existing MapleSim component library, it is possible for users to write their own mathematical models using the Modelica language and custom components. Furthermore, users are able to define properties of components such as parameters and port variables. MapleSim enables users to visualize the model in a 3D environment. A further useful advantage of this software is that users can visualize the simulation results by playing an animation of the motion of the multi-body system. MapleSim templates allow users to analyze the designed models. The model can be linearized, discretized and controllers can be designed for the multi-body system. The response of the systems in time/frequency domain can be analyzed [68].

2.7 Summary

In this chapter, TD and RL methods are reviewed and the gaps in the application are stated which motivates this thesis. Value function and control policy learning equations are derived to provide basis for the rest of the applications in this thesis. Finally, a number of current HRs are introduced, control/learning problems in HR walking are briefly discussed and three multi-body modelling software packages are briefly presented.

3 Value Function Convergence Analysis with Bang-Bang Control

A simple, balance test problem will now be introduced which is used to analyze the residual gradient and TD(0) learning algorithms. A closed form expression for the test problem's terminal value function can be computed; hence the learning convergence of the algorithm and rate of convergence can be studied in detail.

3.1 Test Problem

This section introduces the test problem dynamics that will be used for the convergence and rate of convergence analysis of the value function learning.

3.1.1 Exemplar Test Problem

To simulate and analyze the performance of the RL algorithms, unstable dynamic systems, such as the classic cart-pole, have been used where the objective is to stabilize the system. Consider the (linearized around the upright) dynamics of a point mass at the end of a zero mass pole which is constrained to rotate around its base:

$$ml^2\ddot{x} + mglx = u \quad (21)$$

where g is gravity, u is the applied torque, x is the angular position and m, l are the point mass and pole length, respectively. The unstable mode of this system can be represented as a 1st order system:

$$\tau \dot{x} - x = c_g u \quad (22)$$

with time constant τ and gain c_g . The equation (22) just represents unstable mode of the second order system given by the equation (21). The discrete form of equation (22) is given by:

$$x_{k+1} = ax_k + bu_k \quad (23)$$

where $a (> 1)$ and b are the parameters of the discrete time, 1st order system and k is the discrete time index. It is assumed that, without loss of generality, $|u_k| \leq 1$. This 1st order system has been used to approximate the original system since they have similar transient behaviours and they are used as a model of balance [71]. The time-optimal

control signal, which minimizes the time from the upright, is a bang-bang control action:

$$u_k = -\text{sgn}(x_k) \quad (24)$$

For the purpose of brevity, let $\text{sgn}(x_k) = s(x_k)$ and it is assumed that $s(0) = 0$.

3.1.2 Test Problem Dynamics

As the control signal is bounded, the system can be stabilized when the initial state lies in the region $x_0 \in \{-b/(a-1), b/(a-1)\}$. Two equilibrium points occur at $|x_0| = b/(a-1)$, and the control signal's magnitude is insufficient to stabilize the system when $|x_0| > b/(a-1)$. The behaviour of the equivalent 1st order, continuous time system with optimal bang-bang control is to converge to the upright in a finite time and remain there as the applied control signal is zero when the unstable equilibrium is reached. The bang-bang control signal is called optimal here since it minimizes the state of the first order test problem together with minimizing time to yield a terminal state at the origin. However, for the discrete time system, this is only true for a countable set of points, and more generally, the system will exhibit chaotic behaviour about the upright as described in Figure 6. It is worth noting that the existence of chaotic dynamics implies the existence of multiple unstable limit cycles, and this indeed occurs.

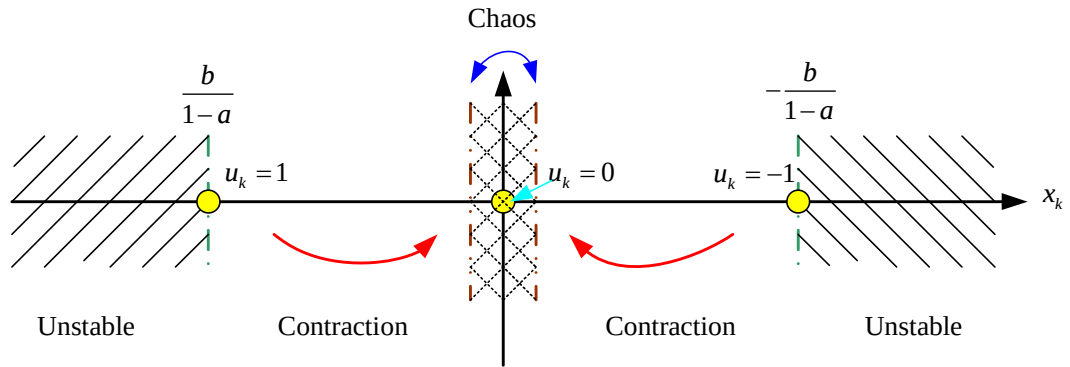


Figure 6 : Dynamics of the system in various regions.

3.1.3 Finite Horizon Dynamics

Consider an initial state x_0 which reaches the unstable upright equilibrium in a finite time and terminates exactly at the origin. The closed form expression for the dynamics is given by:

$$x_k = \frac{b}{a-1} s(x_0) + a^k \left(x_0 - \frac{b}{a-1} s(x_0) \right) \quad (25)$$

Using the fact that the sequence x_k is single signed; hence $s(x_k) = s(x_0)$. This means that the value function learning is considered only positive or negative state regions. However, this assumption does not affect the value function learning outcomes. When the pendulum reaches the unstable equilibrium $x_{k^*} = 0$, then a zero torque is applied and all future states are zero. This happens when:

$$k^* = \left\lceil \frac{\ln \left(-\frac{b}{a-1} s(x_0) \right) - \ln \left(x_0 - \frac{b}{a-1} s(x_0) \right)}{\ln(a)} \right\rceil \quad (26)$$

If k^* is an integer, the initial state converges in a finite time as shown in Figure 7 and as such this only occurs for a countably infinite set of points which can be obtained by reversing the equations of motion.

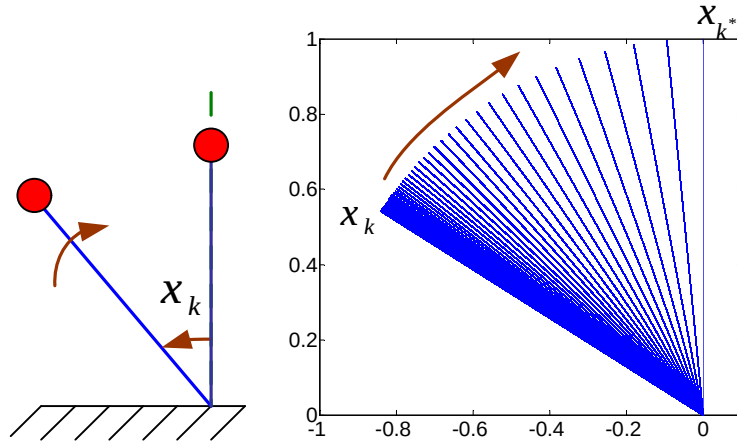


Figure 7: The dynamics of the system where the terminal state is zero.

The simulation parameters are $a = 1.1051$, $b = 0.1051$, $x_0 = -0.99833$ radian and $k^* = 64$ samples. This is a valid argument as long as the assigned initial state yields zero terminal state, otherwise chaotic dynamics described next occur.

3.1.4 Chaotic Terminal Dynamics

Consider now a more general case when the initial state lies in the contracting region, but does not converge to the upright in a finite time where k^* is not an integer and the value function formulation is infinite horizon rather than finite horizon. Then the discrete time system will converge to a bounded region about the upright in which chaotic behaviour occurs [72]. In this case, the bang-bang control signal is not optimal since initial state does not yield zero terminal state at the origin and leads a number of bangs around the origin. This is due to the discrete sampling where the system is applying an incorrect control action for part of the sample period and as such will be exponentially diverging. The behaviour is only corrected at the next sample time which pushes the pole back towards the vertical equilibrium, as illustrated in Figure 8. In this figure, it has been assumed that the discrete time sample is relatively short; hence $a < \sqrt{2}$. When $a > \sqrt{2}$, more complex chaotic dynamics result [73]. However, this is unlikely for an appropriate sampling time as significantly short sampling time leads convergent dynamics.

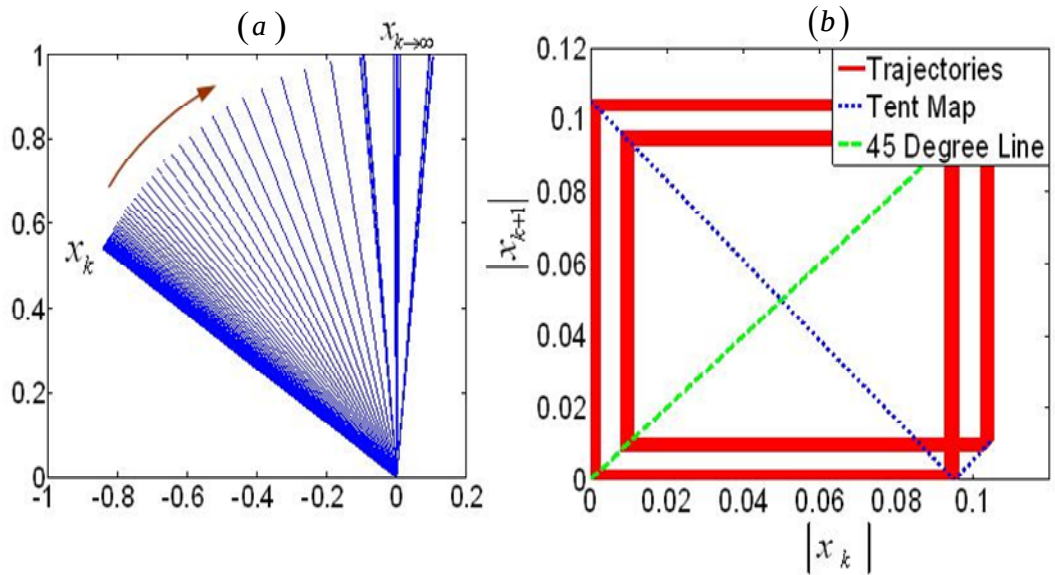


Figure 8: a) Chaotic trajectories of the test problem b) tent map based analysis.

The initial state for the simulation is $x_0 = -0.99$ radian and the rest of the simulation parameters are same as the given in the section 3.1.3. For this test problem analysis, it is assumed that the system does not have chaotic dynamics.

3.2 Analysis of the Value Function Learning for the Test Problem

The existence of a test problem for which the solution is known allows a detailed evaluation of both parameter convergence and the rate of convergence. Therefore, in this section a closed form expression, which has a polynomial form, is derived for the terminal value function of the exemplar balance problem. It is shown that during the state trajectory, there is a non-zero null space associated with the value function's basis. While these are shown for this particular test problem, it is hypothesised that they will, in fact, hold for an arbitrary problem which terminates in a finite time. Solutions to overcome this null space are proposed and the problem of the value function estimation when the discrete time system has a chaotic dynamic behaviour is analyzed.

3.2.1 Finite Horizon Value Function

For the finite horizon terminal value function problem, it is possible to explicitly calculate the value function when the system dynamics are available. The RL value function was introduced in section 2.4.1 and the finite horizon value function with zero terminal reward is given by:

$$V(x_k) = \sum_{t=0}^{k^*} \gamma^t |x_{k+t}| \quad (27)$$

where $r(x_k) = |x_k|$ is the absolute state based reward and k^* is the finite horizon sample. It is important to note that the value function learning with a state dependent reward is performance based rather than stability based. The closed form solution of the terminal value function expressed in section 3.2.2 with this state based reward has bias and linear parameters where both of them occur due to reward signal. On the other hand, while this closed form solution with a binary reward will not have the linear parameter, with a quadratic reward an extra quadratic term will occur together with the bias and linear parameters. Thus, the defined reward signal plays an important role in determining the shape of the value function. However, similar insights can be made for all cases.

3.2.2 Terminal Value Function

In order to examine the learning properties of the TD(0) and residual gradient learning algorithms, a closed form expression for the terminal value function is now derived.

Theorem 3.1: The closed form finite horizon terminal value function solution for the balance test problem is given by:

$$V(x_k) = \left| -\frac{b}{a-1} s(x_k) \right| \frac{1}{1-\gamma} - \left| x_k - \frac{b}{a-1} s(x_k) \right| \frac{1}{1-\gamma a} + \left| -\frac{b}{a-1} s(x_k) \right|^{\ln(\gamma a)/\ln(a)} \left| x_k - \frac{b}{a-1} s(x_k) \right|^{-\ln(\gamma)/\ln(a)} \left(\frac{\gamma a}{1-\gamma a} - \frac{\gamma}{1-\gamma} \right)$$

where the finite horizon sample is an integer for the terminal value function solution and the optimal bang-bang control signal u_k is known yielding zero terminal state.

Proof: The closed form solution for the terminal value function is derived by taking into account the absolute state based reward $r(x_k) = |x_k|$, where x_k is specified by (25). The system converges to zero after k^* iterations; hence, the terminal state is zero and also k^* is an integer. Substituting the absolute state based reward into the value function expression (27) yields:

$$V(x_k) = \sum_{t=0}^{k^*} \gamma^t \left| \frac{b}{a-1} s(x_k) + a^t \left(x_k - \frac{b}{a-1} s(x_k) \right) \right| \quad (28)$$

Substituting the finite horizon k^* from equation (26) and expanding it gives:

$$V(x_k) = \left| -\frac{b}{a-1} s(x_k) \right| \frac{1}{1-\gamma} - \left| x_k - \frac{b}{a-1} s(x_k) \right| \frac{1}{1-\gamma a} + \left| -\frac{b}{a-1} s(x_k) \right|^{\ln(\gamma a)/\ln(a)} \left| x_k - \frac{b}{a-1} s(x_k) \right|^{-\ln(\gamma)/\ln(a)} \left(\frac{\gamma a}{1-\gamma a} - \frac{\gamma}{1-\gamma} \right) \quad (29)$$

The closed form solution of the terminal value function consists of the discount factor, parameters and state of the system.

□

Step by step derivation of the closed form terminal value function solution (29) is given in Appendix A. The piecewise nature of the polynomial terms is due to the instantaneous reward which is the absolute value of the state. The value function has a

constant term, a linear term and a higher order term, where the order and parameters depend on both $\square$ and a . The constant/linear terms in the value function should not be surprising because the instantaneous reward is a linear function of the state. The higher order polynomial term is due to the finite horizon associated with terminating the experiment.

Remark 3.1: For convergent value function parameters, the value function must have $\gamma a < 1$ or $\gamma < 1/a$. In the value function expression, the parameters associated with the linear and higher order parameters approach ∞ as $\gamma \rightarrow 1/a$. Similarly, the power of the higher order basis function approaches 1 and this leads to a singular condition problem. The condition of the learning problem will be poor for $\gamma \approx 1/a$ and this demonstrates the clear relation between the γ discount factor and the time constant of the system.

3.2.3 Optimal Basis and Parameters for the Value Function

The optimal parameter vector of the terminal value function (29) is given by:

$$\mathbf{w}^* = \left[\left| \frac{b}{a-1} s(x_k) \right| \frac{1}{1-\gamma} \quad \frac{1}{1-\gamma a} \quad \left| \frac{b}{a-1} s(x_k) \right|^{\ln(\gamma a)/\ln(a)} \left(\frac{\gamma a}{1-\gamma a} \frac{\gamma}{1-\gamma} \right) \quad \frac{b}{a-1} s(x_k) \quad \frac{\ln(\gamma)}{\ln(a)} \right]^T \quad (30)$$

The k^{th} estimate is represented as $\mathbf{w}_k = [w_{1,k} \quad w_{2,k} \quad w_{3,k} \quad w_{4,k} \quad w_{5,k}]^T$ where w_1 is the bias parameter, w_2 is the linear parameter and w_3 is the higher order parameter of the closed form terminal value function solution, w_4 is the gain of the test problem and w_5 is the power of the higher order parameter. Even though the dynamic system in the exemplar balance problem has just two parameters, the terminal value function has more unknown parameters to estimate when represented in piecewise polynomial form.

The optimal basis of the terminal value function solution is:

$$\phi(x_k, w_k) = \underbrace{|w_{4,k}|}_{\text{Basis for Bias Parameter}} \underbrace{|x_k + w_{4,k}|}_{\text{Basis for Linear Parameter}} \underbrace{|x_k + w_{4,k}|^{w_{5,k}}}_{\text{Basis for Higher Order Polynomial}} \quad (31)$$

The higher order parameter, assuming $\gamma a < 1$, occurs due to the effect of the finite horizon where it is necessary to estimate the power as it depends on the plant parameter a together with the discount factor γ . As shown in (31), the exemplar test problem has a piecewise polynomial basis function. Using the same basis for the value function estimator will mean that it is possible to study parameter convergence as the modelling error/bias is zero. The analysis of modelling error in RL is usually ignored. However, it is well-known in adaptive control that modelling error can cause instability [74].

The task of the TD learning is therefore to estimate the unknown parameters of the value function with this optimal basis. The bias, linear and higher order parameters involve the dynamics of the plant; hence they must be learned.

Two important points are also worth noting 1) in general, the basis signal is a function of the (unknown) plant dynamics hence it is not possible to assume that such an optimal basis is available for a practical application and 2) generally neural networks local basis function are poor at approximating polynomial functions without explicitly modifying their form, such as by the addition of local linear terms. The majority of the RL approaches use local basis functions (RBF, kernels, ...) [9, 13]; hence it is expected that the modelling error may be significant, although this is not addressed in this thesis.

3.3 Trajectory Null Space of the TD Error

In this section, it is shown that the TD error, during the trajectory, has a null space associated with the differenced basis which corresponds to the finite horizon. A number of solutions are proposed to estimate the corresponding parameters at the end of the horizon.

Theorem 3.2: TD error has a non-zero null space along the optimal trajectory as the differenced basis associated with the higher order parameter is:

$$\phi(x_k, w_{3,k}) - \gamma \phi(x_{k+1}, w_{3,k}) = 0$$

where $\phi(x_k, w_{3,k})$ is the basis for the higher order parameter.

Intuitive Proof: Assume that the higher order parameter is $w_{3,k} \neq 0$ and the corresponding differenced basis is $\phi(x_k, w_{3,k}) - \gamma\phi(x_{k+1}, w_{3,k}) \neq 0$. Then, the estimated reward which is given by:

$$\hat{r}(x_k, w_k) = (\phi(x_k, w_k) - \gamma\phi(x_{k+1}, w_k)) w_k^T$$

spans in the parameter space with the bias (b), linear (l) and higher (h) order parameters.

$$\hat{r}(x_k, w_k) = \langle b, l, h \rangle$$

As discussed in section 3.2.1, the instantaneous reward span in the parameter space with the bias and linear parameters.

$$r(x_k) = \langle b, l \rangle$$

In this case, the TD error which has representation as:

$$\delta_k = r(x_k) - \hat{r}(x_k, w_k)$$

is non-zero. Thus, the differenced higher order basis must be:

$$\phi(x_k, w_{3,k}) - \gamma\phi(x_{k+1}, w_{3,k}) = 0$$

in order to yield zero TD error. This shows that the higher order parameter cannot be learned until the terminal point.

□

Proof: The optimal polynomial basis function given by (31) is used for the null space analysis of the differenced basis function. Here $w_{4,k}$ and $w_{5,k}$ parameters are assumed to be known and $\Phi(x_k, w_k)$ represents the differenced basis $(\phi(x_k, w_k) - \gamma\phi(x_{k+1}, w_k))$.

$$\begin{aligned} \Phi(x_k, w_k) &= (\phi(x_k, w_k) - \gamma\phi(x_{k+1}, w_k)) \\ &= \left(\begin{bmatrix} 1 & |x_k + w_{4,k}| & |x_k + w_{4,k}|^{w_{5,k}} \end{bmatrix} - \gamma \begin{bmatrix} 1 & |x_{k+1} + w_{4,k}| & |x_{k+1} + w_{4,k}|^{w_{5,k}} \end{bmatrix} \right)^T \\ &= \left(\begin{bmatrix} 1 & |x_k + w_{4,k}| & |x_k + w_{4,k}|^{w_{5,k}} \end{bmatrix} - \gamma \begin{bmatrix} 1 & |ax_k - bs(x_k) + w_{4,k}| & |ax_k - bs(x_k) + w_{4,k}|^{w_{5,k}} \end{bmatrix} \right)^T \end{aligned} \quad (32)$$

Since $-bs(x_k) + w_{4,k} = -bs(x_k) - bs(x_k)/(a-1)$ and this is equal to $abs(x_k)/(a-1)$ which is basically $aw_{4,k}$. Therefore (32) becomes:

$$\begin{aligned}\Phi(x_k, w_k) &= \left(\begin{bmatrix} 1 & |x_k + w_{4,k}| & |x_k + w_{4,k}|^{w_{5,k}} \end{bmatrix} - \gamma \begin{bmatrix} 1 & |ax_k + aw_{4,k}| & |ax_k + aw_{4,k}|^{w_{5,k}} \end{bmatrix} \right)^T \\ &= \left(\begin{bmatrix} (1-\gamma) & |x_k + w_{4,k}|(1-\gamma a) & |x_k + w_{4,k}|^{w_{5,k}} \left(1 - \gamma \left(e^{\ln(a)} \right)^{w_{5,k}} \right) \end{bmatrix} \right)^T\end{aligned}\quad (33)$$

Finally, substituting $w_{5,k} = -\ln(\gamma)/\ln(a)$, which is the power of the higher order parameter, in (33) becomes:

$$\Phi(x_k, w_k) = \left(\begin{bmatrix} (1-\gamma) & |x_k + w_{4,k}|(1-\gamma a) & |x_k + w_{4,k}|^{w_{5,k}} \left(1 - \gamma \frac{1}{\gamma} \right) = 0 \end{bmatrix} \right)^T \quad (34)$$

The third column which is the basis associated with the higher order parameter is zero.

□

Therefore, both the expected and instantaneous rewards span in the parameter space without carrying any information about the higher order polynomial until reaching the end of the batch/horizon. This result is not surprising since the higher order parameter is due to finite horizon effect; hence it only appears at the finite horizon sample k^* .

Remark 3.2: It is important to note that the value of the higher order basis exponentially increases as the state x_k approaches zero. Furthermore, the higher order parameter has more effect on the value function as $x_k \rightarrow 0$. This insight is valid for the linear parameter but not the bias parameter as its basis is not state dependent. Based on the non-zero null space analysis introduced by the theorem 3.2, two important corollaries are now given.

3.3.1 Trajectory Null Space Effect on Residual Gradient Learning

Obviously, both the residual gradient and TD(0) algorithms explicitly depend on the TD error. Therefore, the instantaneous estimation of the higher order parameter $w_{3,k}$ will be poor as the TD error is independent of this parameter until the terminal state is reached. The corollary 3.1 addresses this case for the residual gradient algorithm.

Corollary 3.1: The higher order parameter update is zero for the residual gradient learning algorithm until the terminal state is reached.

Solution: Consider the residual gradient learning parameter update rule introduced in section 2.4.4 where the search direction is proportional to the differenced basis function.

$$\Delta w_{1:3,k} = \eta \left(r(x_k) - (\phi(x_k, w_k) - \gamma \phi(x_{k+1}, w_k)) w_{1:3,k}^T \right) (\phi(x_k, w_k) - \gamma \phi(x_{k+1}, w_{k+1})) \quad (35)$$

Theorem 3.2 proves that the differenced basis associated with the TD error has a non-zero null space. Therefore, the TD error does not carry any information about the higher order parameter. Moreover, the differenced basis occurring due to gradient of the differenced value function has no information about this higher order parameter; hence $\Delta w_{3,k} = 0$ for the residual gradient learning.

□

The simulation result presented later in section 3.6 justifies that there is no update for this particular parameter during the state trajectory as well. As theorem 3.2 and corollary 3.1 stated that the TD error does not contain any information about the higher order parameter, $w_{3,k}$, however, this is not the case for TD(0).

3.3.2 Impact of the Trajectory Null Space on the TD(0) Learning

The non-zero null space has a different effect on the higher order parameter learning for TD(0) due to difference in formulation as the corollary 3.2 expresses.

Corollary 3.2: The higher order parameter update is non-zero for the TD(0) algorithm until the terminal state is reached.

Solution: The TD error and hence the TD(0) learning algorithm described in section 2.4.3 is independent of the higher order parameter, $w_{3,k}$, during the state trajectory. The higher order parameter, $w_{3,k}$, does not affect the TD(0) because it is multiplied by the zero, differenced basis function. However, because of the formulation of the TD(0) update, the parameter is still updated because the update rule is based on the basis function vector at time k , not the difference.

$$\Delta w_{13,k} = \eta \left(r(x_k) - (\phi(x_k, w_k) - \gamma \phi(x_{k+1}, w_k)) w_{13,k}^T \right) \phi(x_k) \quad (36)$$

While this causes non-zero weight updates, $\Delta w_{3,k} \neq 0$, this should be regarded as parameter drift during the trajectory because the error is independent of this parameter.

□

In general, this null space, which will cause both the residual gradient and the TD(0) learning algorithms to converge to biased solutions, can be overcome using several different, but equivalent conditions. In order to be able to estimate $w_{3,k}$, extra information must be constructed for both residual gradient and TD(0) learning at the terminal sample of the finite horizon as discussed next.

3.4 Estimating the Higher Order Parameter

As expressed in theorem 3.2, the TD error has a non-zero null space until the terminal point is reached. Three different approaches can be applied to learn the higher order polynomial by taking into account the information at the terminal sample.

- 1) TD learning is applied for the point $x_{k^*} = x_{k^*+1} = 0$ where the control signals are $u_{k^*-1} = 1$ and $u_{k^*} = 0$. In this case, the corresponding basis is formed as:

$$\phi(x_{k^*}, w_{k^*}) = \begin{bmatrix} w_{4,k^*} & w_{4,k^*} & w_{4,k^*} \end{bmatrix}^T \quad (37)$$

where w_{4,k^*} is unity in this case.

- 2) Consider the terminal condition $\hat{V}(x_{k^*}, w_{k^*}) = r(x_{k^*}) = 0$ and update the parameter vector by just using this available information. So the new basis is just in terms of the $k^* - 1$ state.

$$\phi(x_{k^*-1}, w_{k^*-1}) = \begin{bmatrix} w_{4,k^*-1} & x_{k^*-1} + w_{4,k^*-1} & x_{k^*-1} + w_{4,k^*-1} \end{bmatrix}^{w_{5,k^*}}.$$

- 3) A constraint $1w_{k^*}^T = 0$ is imposed during the value function learning, since the sum of the parameters is equal to zero for $\hat{V}(x_{k^*}, w_{k^*}) = 0$. Hence, $-w_{3,k} = w_{1,k} + w_{2,k}$ for the terminal sample.

Figure 9 configures the parameter learning at the last sample of the horizon.

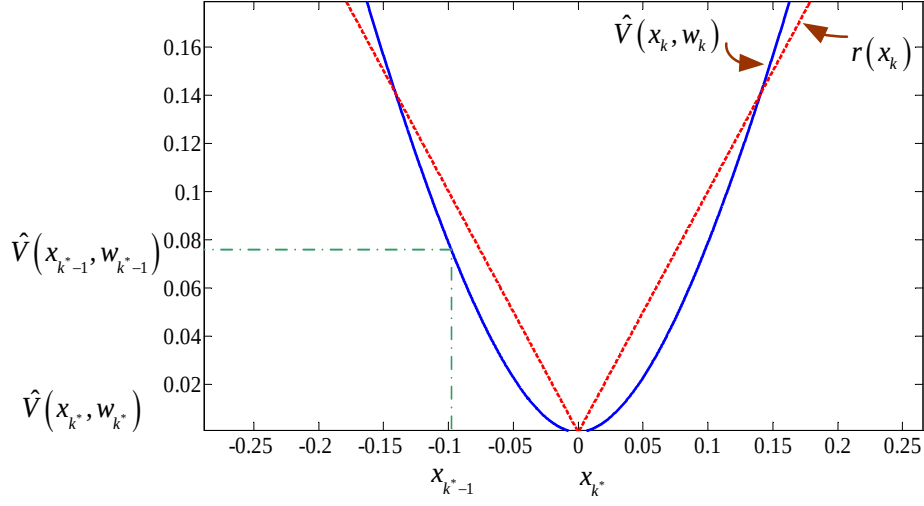


Figure 9: Configuration of the higher order polynomial learning at the final horizon.

These conditions are equivalent and yield unbiased parameter convergence. So far, it has been assumed that the initial state is such that the states converge to zero after a finite time [75]. The next section discusses how the insights gained so far can be adapted to the case for other initial states which converge to a chaotic attractor about the upright.

3.5 Value Function Learning with Chaotic Dynamics

In general, the discrete time, balance system converges to a chaotic attractor about the upright. A small amount of offset due to digital chaos occurs in the value function representation.

$$V(x_k) = \sum_{t=0}^{\infty} \gamma^t r(x_{k+t}) + \varepsilon(x_k) \quad (38)$$

Here $\varepsilon(x_k)$ represents the digital offset in the value function for each state sample. This digital chaos causes a problem with the RL analysis as the closed form solution for the value function is not easy to obtain. Although approximations can be made, as will now be described, much of the insights obtained in the previous sections can be applied to this case.

The initial state contracts from its initial value until k_d^* which is the sample to take the initial state to the edge of the chaotic region and then starting from $k_d^* + 1$ to infinity the states lie on a bounded chaotic attractor. The value function can be represented as:

$$V(x_k) = \sum_{t=0}^{k_d^*} \gamma^t r(x_{k+t}) + \sum_{t=k_d^*+1}^{\infty} \gamma^t r(x_{k+t}) \quad (39)$$

The first part of the value function expression (39) is the same as the closed form terminal value function solution derived in section 3.2.2. The only difference is that the state at k_d^* are non-zero whereas it is zero at k^* in 3.2.2. Thus, the first part has a terminal value function solution having optimal polynomial basis and parameters. However, the second part of (39) covers a region where the chaotic dynamics occur. Since chaotic dynamics are sensitive to initial conditions, obtaining a closed form solution for the value function is not easy. Therefore, a dead-zone can be defined to switch off the learning in the chaotic region so the reward $r(x_k) = 0$ from $k_d^* + 1$. The value function with defined dead zone is:

$$V(x_k) = \sum_{t=0}^{k_d^*} \gamma^t r_d(x_{k+t}) \quad (40)$$

where $r_d(x_k) = |x_k| - |x_{k_d^*}|$ is the instantaneous reward with a designed dead zone.

Since the solution for (40) has a higher order parameter, the described theorem in section 3.3 for the null space associated with the TD error and residual gradient/TD(0) parameter updates hold for this case as well. The next section describes simulations to analyze the rate of convergence and parameter convergence for both residual gradient and TD(0) learning. Also, it presents value function graphs with chaotic dynamics and designed dead-zone.

3.6 Balance Test Problem Simulation

In this section, the parameter convergence associated with the terminal value function for the exemplar balance problem is analyzed. For simulation Matlab software is used. The relationship to the traditional LMS algorithm is emphasized by illustrating how the data is projected onto the differenced basis function vector in the parameter space. The null space associated with the TD error is illustrated and the rate of convergence

of the residual gradient and the TD(0) learning algorithms are analyzed and compared. Finally, the value function learning in the case of having chaotic dynamics is addressed.

3.6.1 Simulation Details

The discrete time, exemplar balance control problem, described in the section 3.1, is given by:

$$x_{k+1} = 1.1051x_k - 0.1051s(x_k) \quad (41)$$

This can be obtained from the continuous time system by assuming a sample time of 0.1s with unity parameter values. Therefore, the test problem parameters are $a = 1.1051$ and $b = 0.1051$. Without loss of generality, the initial state is assumed to be negative and the corresponding optimal control action is unity (until the state is driven to zero). The simulation consists of 390 batches where each batch has initial state to $x_0 = -0.99833$ and running for $k^* = 64$ iterations until the pole reaches the upright position.

The terminal value function uses the optimal polynomial basis given in (31) which assumes that $w_{4,k}$ and $w_{5,k}$ parameters are known and the remaining parameters are to be estimated. While this is an unreasonable assumption, it simplifies the analysis of parameter convergence performed in this chapter. The gain of the system is 1, so $w_{4,k} = 1$. The discount factor is $\gamma = 0.85$ with an upper bound of ~ 0.9 for the value function sum to converge. Therefore, the terminal value function is:

$$V(x_k) = 6.66 - 16.50|x_k + 1| + 9.83|x_k + 1|^{1.62} \quad (42)$$

Hence, the optimal polynomial basis is:

$$\phi(x_k) = \begin{bmatrix} 1 & |x_k + 1| & |x_k + 1|^{w_{5,k}} \end{bmatrix}^T \quad (43)$$

The simulation starts with a zero initial parameter vector.

Learning stability is assured if the learning rates lie in the intervals $0 < \eta^{RG} < 29.6$ and $0 < \eta^{TD(0)} < 4.44$ (where RG stands for residual gradient). The learning rate intervals are specified for stable learning in the sections 2.4.3 and 2.4.4, for RG and

TD(0) respectively. The maximum value of the basis $\|\phi(x_k) - \gamma\phi(x_{k+1})\|_2^2$ is 0.067 for the RG while the maximum value of the basis $(\phi(x_k) - \gamma\phi(x_{k+1}))^T \phi(x_k)$ is 0.45 for the TD(0) algorithm. These maximum values occur when the initial state approaches the origin. Furthermore, it is clear that $\|\phi(x_k) - \gamma\phi(x_{k+1})\|_2^2 < (\phi(x_k) - \gamma\phi(x_{k+1}))^T \phi(x_k)$ so to compare the convergence rates, the learning rates should be chosen to reflect this. In this simulation, the learning rates are set to be half of their maximum values $\eta^{RG} = 14.8$ and $\eta^{TD(0)} = 2.22$. Obviously, when there exists significant noise/modelling error, the learning rates will have to be considerably smaller. Now the task is to estimate these parameters using the defined optimal polynomial basis.

3.6.2 Parameter Convergence

The convergence of the residual gradient and the TD(0) learning algorithms is now compared. As can be seen in Figure 10, there is a repeated zigzag pattern in the bias, w_1 , and linear, w_2 , parameters where each zigzag represents a batch consisting of 64 samples. Initially, the linear term associated with the differenced basis function vector is near zero and the parameters are projected horizontally towards the solution line (set of parameter values that satisfy that datum). As the system rotates towards the upright equilibrium, the linear term in the differenced basis function vector becomes significant and the solution hyper plane rotates in an anti-clockwise direction, until it lies at an angle of about 45° with the upright. During the experiment, the parameter estimates are projected towards the rotating solution hyperplane as shown in Figure 10. This is similar for both the residual gradient and the TD(0) algorithms.

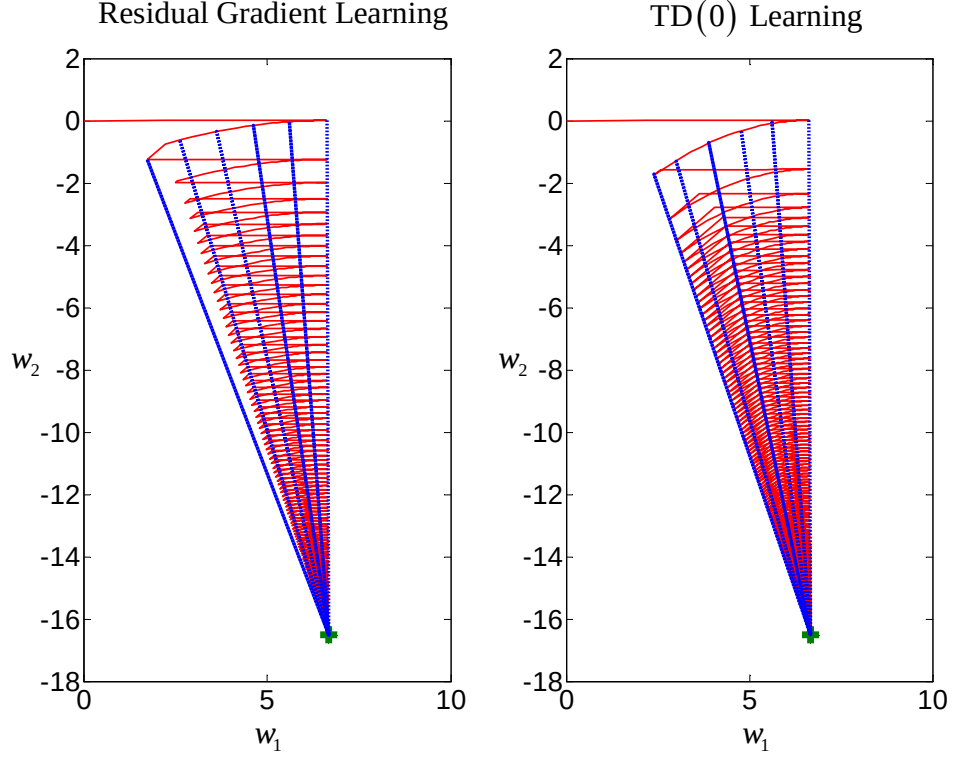


Figure 10: The parameter convergence for the bias w_1 and linear w_2 parameters of the residual gradient and TD(0) learning.

The solid red line shows the parameter trajectories and dotted blue line represents the solution lines associated with each training datum (corresponding to the differenced basis functions). It is important to note that at the end of the each batch, the final parameters for the residual gradient learning do not reach the same solution line due to the fact that the parameters are not totally independent of w_3 . However, this is not the case for TD(0) as updates to w_3 are non-zero, TD error is independent of w_3 . It is also noticeable that the shape of the parameter trajectories of the residual gradient and TD(0) learning show small differences due to having different parameter update rules [76]. This difference is more significant for the higher order parameter learning as discussed next.

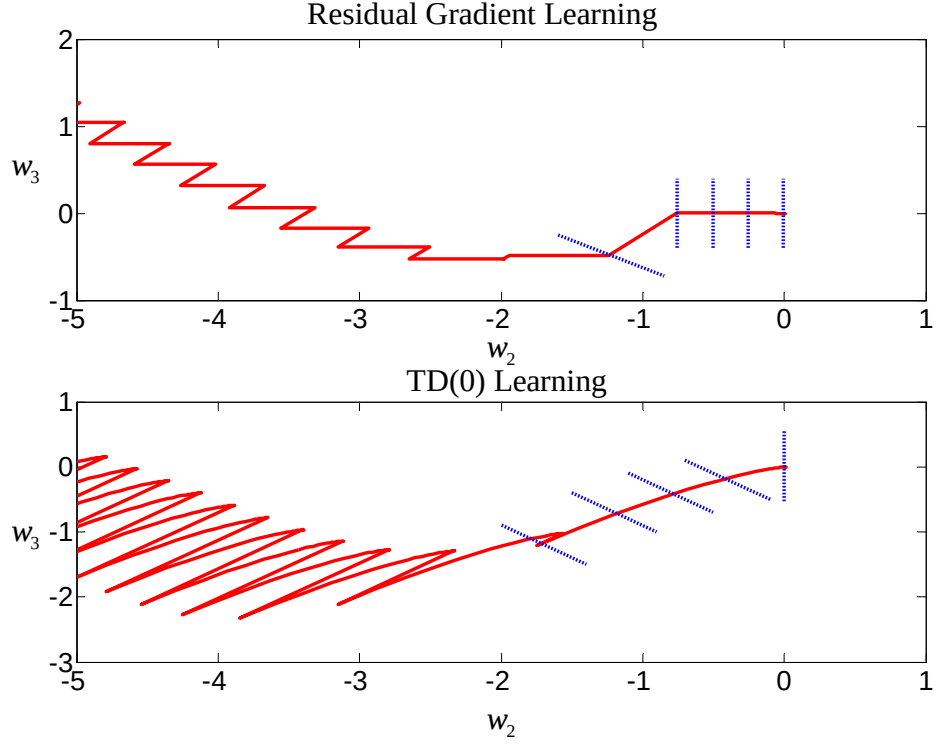


Figure 11: The parameter convergence of the linear w_2 and higher order parameter w_3 of the residual gradient and TD(0) learning.

The red curves represent parameter trajectories and dotted blue lines are solution lines (shown just for the first batch). Figure 11 shows the initial trajectory associated with estimating the linear, w_2 , and higher order, w_3 , parameters. The convergence of w_3 differs in that for the whole of the experiment, the corresponding residual gradient update is zero and it remains unchanged. For the terminal point, where the pendulum remains stationary in the upright position, the solution hyperplane is rotated and the parameter vector is projected onto it. Then the simulation is repeated and the parameter update trajectory is repeated. For TD(0), w_3 has a non-zero update when the pendulum is rotating towards the upright. However, it should be observed that it appears to be drifting away from the true value. The TD error only depends on w_3 at the terminal point, when the pendulum remains stationary and upright, as described in section 3.1.3 [76].

3.6.3 Rate of Convergence

The rate at which the parameter estimates approach their true values is important as the system should efficiently use limited simulation resources and/or be used for designing larger, more complex systems. The parameter convergence (norm of the parameter error) is shown in Figure 12 for both the residual gradient and TD(0) learning algorithms.

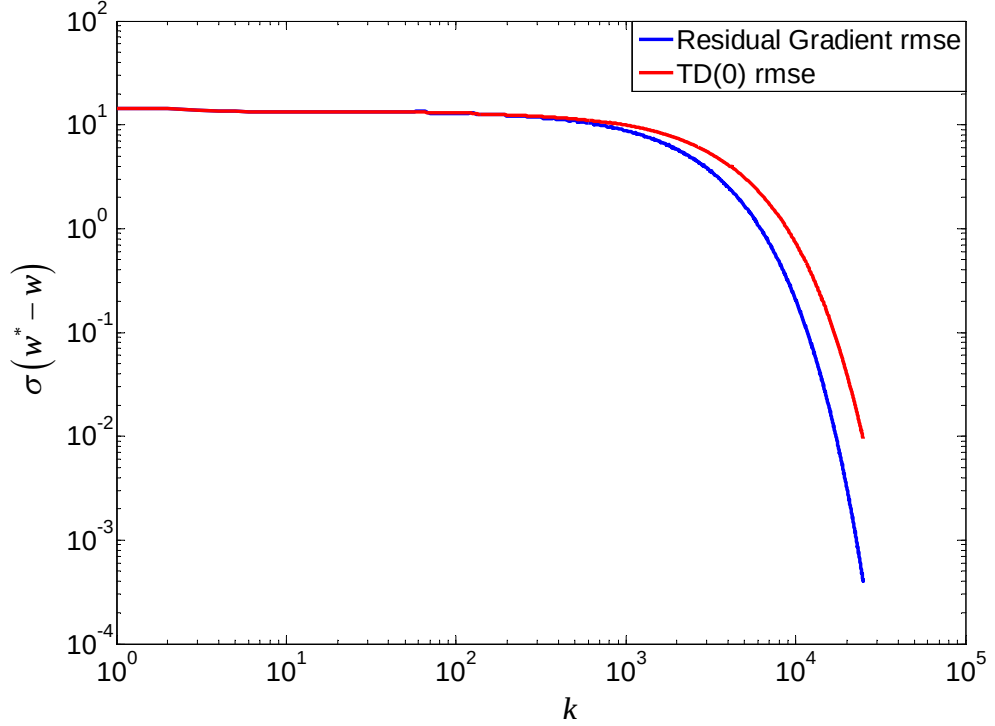


Figure 12: RMS parameter error comparison of the residual gradient and TD(0) learning.

Here k represents the samples and $\sigma(w^* - w)$ is the RMS error between the correct and estimated parameters. It is clear from Figure 12 that the residual gradient learning converges at a faster rate asymptotically. As discussed above, the bias and linear parameters of the residual gradient learning algorithm are learnt during the whole batch whereas the parameter with the higher order is learnt just at the final sample of the horizon when the pendulum remains stationary in the upright equilibrium. As a result of this property, there is a sharp reduction in RMS error of the value function parameters at the end of the each batch (difficult to see on the graph as all the 390 batches are shown). For TD(0) however, even though all parameters are updated at

each time sample, convergence is slower and it was noted that the higher order parameter seemed to be drifting away within each batch.

To quantify the rate of convergence, condition number and time constant of the corresponding bases can be considered. The condition number of the bases are 2945 and 4829 for the residual gradient and TD(0) algorithms respectively. Thus, it is clear that the TD(0) algorithm has badly conditioned basis and this leads to slower convergence. Same insight can be obtained from the time constants of the learning algorithms as well. The time constants are $-1/\log(1-\eta^{RG}\sigma_{\min}^{RG})=10.78$ for residual gradient and $-1/\log(1-\eta^{TD(0)}\sigma_{\min}^{TD(0)})=18.33$ for the TD(0) algorithms respectively, where $\sigma_{\min}^{RG}$ and $\sigma_{\min}^{TD(0)}$ represent the minimum eigenvalues of the bases associated with the residual gradient and TD(0) algorithms. Since smaller time constant yields faster convergence, the residual gradient algorithm converges faster than the TD(0) algorithm [77].

Often, a large value function discount factor, $\gamma = 0.95$ or 0.98 , is reported in the literature. However, badly conditioned learning problems were observed as $\gamma \rightarrow 1/a$, which is the upper limit for the infinite value function sum to converge. This is shown in Figure 13 for the residual gradient algorithm, although similar effects were noticed for TD(0).

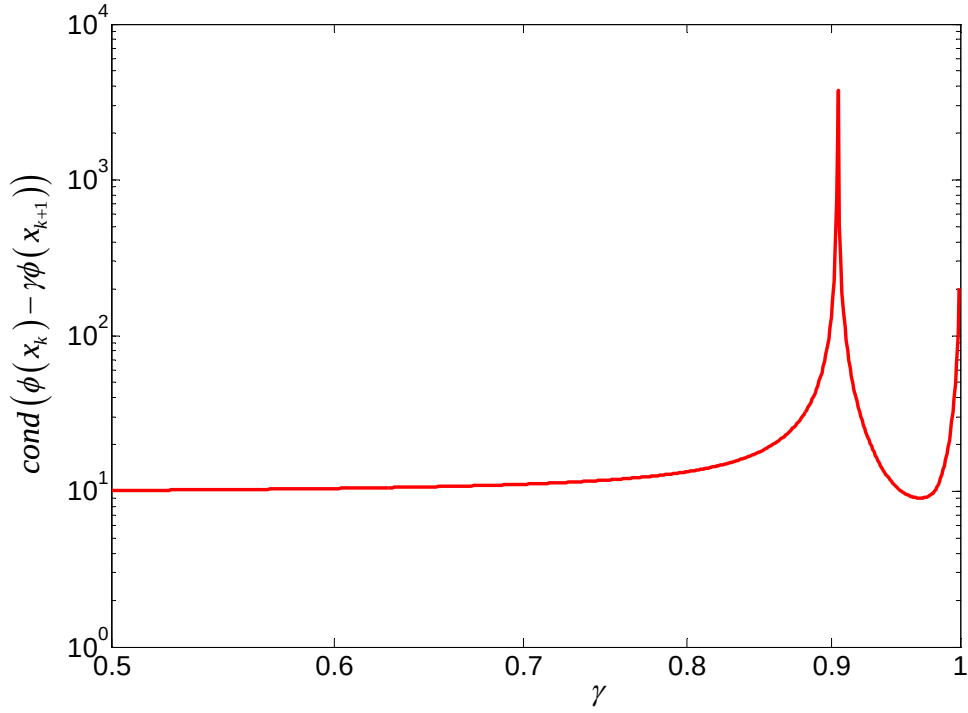


Figure 13: Condition number of the regression matrix plotted against discount factor.

As shown in Figure 13, there are two peaks in the condition number occurring around $1/a$ and 1, where both of them cause a singular basis. If the discount factor is $\gamma = 1/a$, then the power of the basis $|x_k + w_{4,k}|^{-\ln(\gamma)/\ln(a)}$ is 1. Hence, this basis associated with the higher order parameter becomes basically a linear basis which is exactly same as the linear basis for the linear parameter. Furthermore, this will introduce singularity with the linear term. On the other hand, it is important to note that if the discount factor is $\gamma = 1$, then the basis for the higher order parameter returns to basis for the bias parameter and introduces singularity with the bias term. Henceforth, both cases cause badly conditioned basis function. It is important to note that the condition number of the differenced basis has a concave curve around $\gamma = 0.95$ as this discount factor yields power of 0.5 for the higher order basis [75].

3.6.4 Chaotic Dynamics and Value Function Learning

The problem of estimating the value function when the initial state is such that the states converge to a chaotic attractor around the upright will now be analyzed. To determine the value function, consider the infinite horizon value function

representation given by equation (1). In terms of using the test problem (71) representation 300 samples are generated for each initial state ranging from -1 radian to 1 radian for the value function calculation. Note that in each batch there are 64 samples; hence 300 samples take considerably longer time than the time taken to reach the upright, as described in the previous sections. Thus, majority of the states evolve in the chaotic region. The rest of the parameters are the same as described in section 3.6.1.

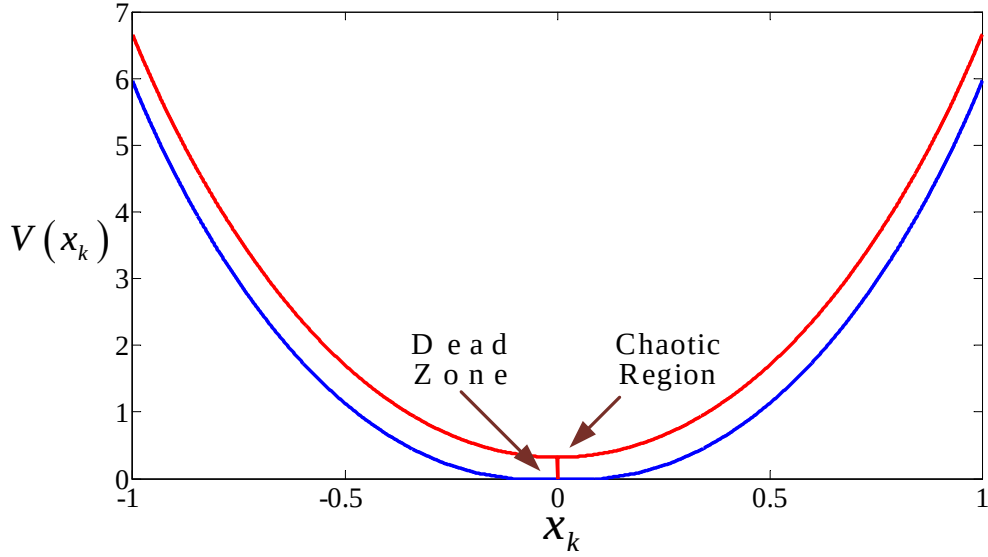


Figure 14: Value function with digital offset and designed dead zone.

As can be seen from the Figure 14, the value function is almost linear outside the chaotic region and there is a digital offset occurring due to the chaos. The rewards are exponentially weighted so the initial states in the chaotic region are relatively more important. The value of this offset can also be estimated/calculated.

Assume that the initial state is $x_0 = 0.1051$, which is the state at the edge of the chaotic region, then the absolute value of the reward is either x_0 which is equal to b or 0. Therefore, the corresponding offset:

$$V(0.1051) = 0.1051(1 + 0.85^2 + 0.85^4 + 0.85^6 + \dots) \approx 0.323 \quad (44)$$

This is the offset value occurring due to digital chaos if the initial state is 0.1051. Furthermore, if the initial state is slightly perturbed from 0, then the corresponding offset is:

$$V(0) = 0.1051(0.85 + 0.85^3 + 0.85^5 + 0.85^9 + \dots) \approx 0.377 \quad (45)$$

This is the offset for the initial state around 0. These results can also be confirmed from Figure 14 as well.

A dead-zone can be introduced for the chaotic region in order to eliminate the effects of chaotic dynamics. Basically, if the dynamics of the system lie in this region, then the learning is switched off and instantaneous reward is zero. Thus, the corresponding value function is zero in the chaotic region as described in Figure 14. The new state based reward is designed as $r_d(x_k) = |x_k| - |x_{k_d}^*|$ where $x_{k_d}^* = b$ is the edge of the chaotic region for this case. It is important to note that the value function has a different behaviour due to characteristic change in instantaneous state based reward, i.e., the maximum value of the value function has changed [75, 77].

3.7 Summary

In this chapter, a simplified balance test problem was introduced which can be used as a test bed for evaluating the value function's parameter convergence by using TD learning. For the test problem, the optimal value function and its corresponding polynomial basis were explicitly calculated. The effect of the finite horizon on the value function parameter estimation was analyzed and it was shown that this introduced a near singularity/bias in the parameter convergence. Three approaches were proposed to address this problem. Even though the proposed methods for learning of higher order polynomial works, the corresponding convergence process is slow. Thus, other higher order learning approaches should be considered. Moreover, it is shown that digital offset appears as a result of having digital chaos. Furthermore, the amount of the digital offset is calculated in terms of using the bounds of chaos and the corresponding discount factor. To eliminate this digital chatter, a dead-zone is defined.

4 Infinite Horizon Value Function Learning Analysis with Piecewise Linear Control

In this chapter of the thesis a closed form solution for the terminal value function of the exemplar balance problem with piecewise linear control is presented. It is expressed that the corresponding terminal value function solution consists of an optimal polynomial basis and parameters. Analysis on the value function parameter convergence with this known optimal basis shows that there is a null space associated with the differenced basis function up to the cut off sample c . Moreover, further analysis proves that the basis is badly conditioned as c gets closer to zero and the discount factor γ approaches $1/a$. Since this differenced basis function is part of the TD error and also residual gradient learning, there is lack of learning for the corresponding parameter until the saturation threshold is reached.

4.1 Test Problem Description

In this section, dynamics of a simple balance test problem with a bounded piecewise linear control is expressed. Infinite horizon discounted value function with a quadratic state based reward analysis in section 4.3 and the performed simulation in section 4.5 use this test problem dynamics. Using the test problem dynamics leads to a closed form solution for the terminal value function; hence the value function parameter convergence and rate of convergence can be studied in detail. This section also introduces saturated piecewise linear control signal and discusses the impact of the control signal on the test problem dynamics. Finally, a Linear Quadratic Regulator (LQR) type reward which is in terms of the system dynamics and control signal is expressed for the infinite horizon value function analysis.

4.1.1 Exemplar Test Problem and Saturated Piecewise Linear Control

To examine the convergence properties of the value function, consider the first order unstable dynamic system introduced in the previous chapter which is given by:

$$\tau \dot{x} - x = c_g u \quad (46)$$

with time constant τ and gain c_g . The discrete form of the equation (22) is:

$$x_{k+1} = ax_k + bu_k \quad (47)$$

where $a (> 1)$ and b are the parameters of the discrete time, 1st order system, k is the discrete time index and u_k is the control signal. The parameters with sampling time $h = 0.1$ are $a = 1.1051$ and $b = 0.1051$.

For this test problem, a saturated piecewise linear control is considered. While bang-bang control used for the previous value function analysis in chapter 3 minimizes the time/state error, the linear piecewise control minimizes the state error from the upright position by requiring less energy. Bang-bang control is not robust as it might yield undesired chaotic dynamics when the bang-bang control is not optimal. Furthermore, controllers such as LQR and saturated piecewise linear, which generate smoother control trajectories, are considered. Sigmoid functions, such as logistic and $\tanh()$, are widely used as a saturation function in RL policy gradient. However, closed form value function solution cannot be obtained with these sigmoid functions. Thus, their less smooth version which is bounded piecewise linear control is considered for this analysis. This kind of control yields realistic learning results. Figure 15 shows the piecewise linear control signal.

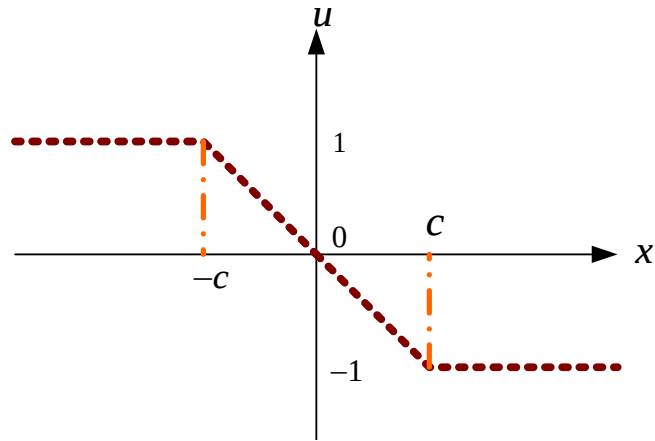


Figure 15: Piecewise linear control, where c is the saturation threshold.

The piecewise linear control signal is defined as:

$$u = \begin{cases} -x/c & |x| < c \\ -\text{sign}(x) & \text{otherwise} \end{cases} \quad (48)$$

Here it is assumed that the control sign is correct, $c > 0$ and for the purpose of brevity $\text{sgn}(x_k) = s(x_k)$ will be used for the remainder of this chapter. It is important to note that as c approaches 0, then the piecewise linear control basically approaches optimal bang-bang control. This might cause chaotic dynamics as described in the next section.

4.1.2 Test Problem Dynamics with Saturated Piecewise Linear Control

With a bounded control signal, the system can be stabilized if the initial state lies in the region $x_0 \in \{-b/(a-1), b/(a-1)\}$, as discussed in chapter 3. Two equilibrium points occur when $|x_0| = b/(a-1)$, and the control signal's magnitude is insufficient to stabilize the system when $|x_0| > b/(a-1)$. Hence, for the remainder of this chapter, it is assumed that the system is stabilizable; otherwise the learning problem would not be well-posed. When the initial state can be stabilized, the system may converge to zero, as would occur with the original continuous time system. However, section 3.1.4 of this thesis shows that this system exhibits bounded chaotic dynamics with non-optimal bang-bang control unless the initial state yields a terminal state at zero. A similar argument is valid for the piecewise linear control case as well if c is inside a region around zero as described in Figure 16.

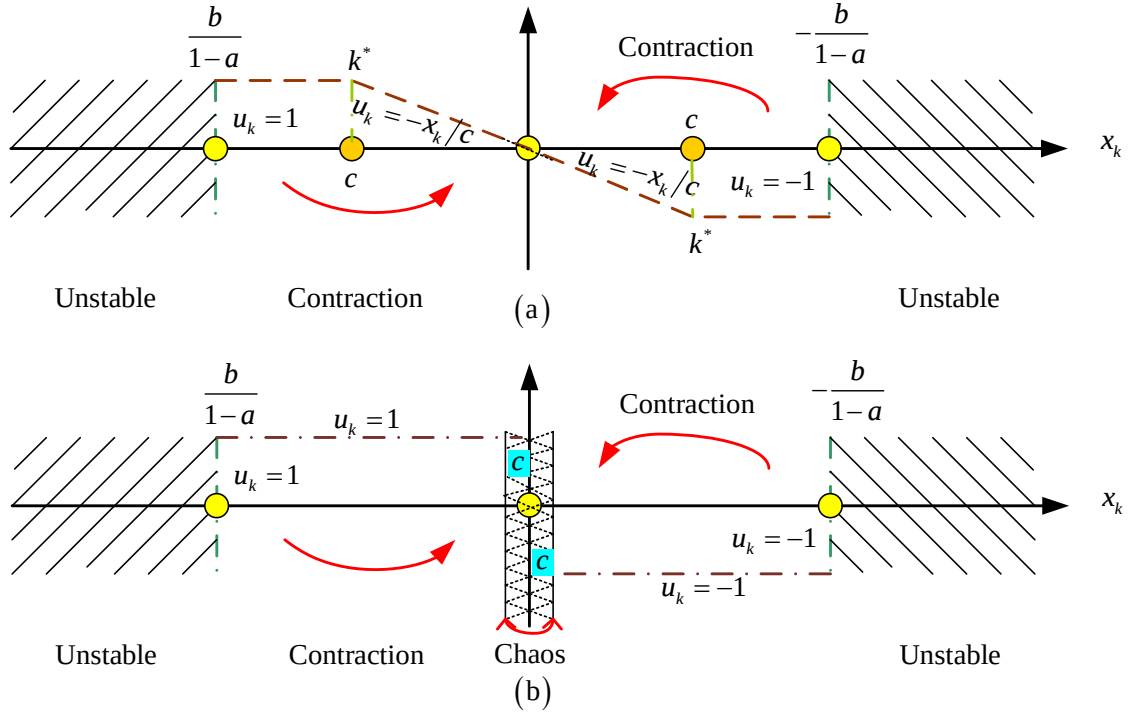


Figure 16: Dynamics of the system in various regions with piecewise linear control, a) convergent dynamics, b) chaotic dynamics around zero.

Here sample k^* is an integer which takes an initial state to the c saturation threshold. Basically, this integer sample assumption is only valid for certain initial states. When the initial state lies in the contracting region, but does not converge to the upright, then the discrete time system will converge to a bounded region about the upright in which chaotic behaviour occurs. Basically the bounds of the chaotic behaviour depend on the discrete sampling time.

Figure 17a describes that an initial state inside the stability region contracts and then converges to zero with the saturated piecewise linear control where c is large enough. However, as the cut off point approaches zero, then initially the system exhibits oscillatory but convergent terminal dynamics as illustrated by Figure 17b. On the other hand, chaotic dynamics occur for smaller cut off value as shown in Figure 17c. Existence and role of chaotic dynamics on value function learning are being addressed in the chapter 3.

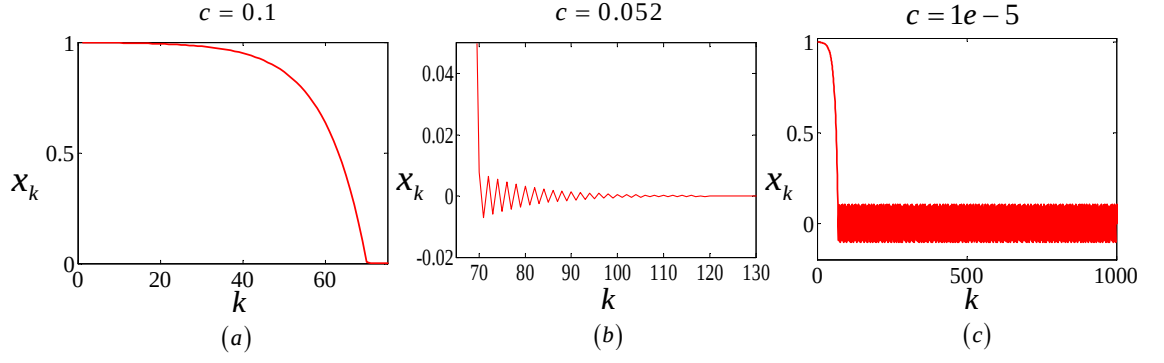


Figure 17: Various dynamical behaviours of the test problem, a) convergent dynamics, b) oscillatory but convergent dynamics, c) chaotic dynamics.

It is clear that the cut off parameter c plays an important role to shape the dynamical behaviour of the test problem [78]. This cut off parameter and its limits for the convergent dynamics can be determined in terms of the system dynamics as described next.

4.1.3 Convergence to the Upright

The basic behaviour of the equivalent 1st order, continuous time system with optimal piecewise linear control is to converge to the upright and remain at the unstable equilibrium as the applied control signal is zero when the unstable equilibrium is reached. This is a valid argument for a discrete time system if the cut off sample is sufficiently large, so that the terminal dynamics are convergent rather than chaotic. The closed form expression for the test problem when $|x_0| \geq c$, so that the initial state is outside the linear region, is given by:

$$x_k = \frac{b}{a-1} s(x_0) + a^k \left(x_0 - \frac{b}{a-1} s(x_0) \right) \quad (49)$$

using the fact that the sequence x_k is single signed; hence $s(x_k) = s(x_0)$. This closed form expression is only valid for the case shown in Figure 17b). When the pendulum reaches the saturation point c , the control signal shifts to linear control from unity.

$$x_{k^*} = c = \frac{b}{a-1} s(x_0) + a^{k^*} \left(x_0 - \frac{b}{a-1} s(x_0) \right) \quad (50)$$

where k^* is the sample in which the state $x_k = c$. The general solution of the k^* for both negative and positive state regions is given by:

$$k^* = \frac{\ln(|w_0| - c)}{\ln(a)} - \frac{\ln|x_0 - w_0|}{\ln(a)} \quad (51)$$

where $w_0 = bs(x_0)/(a-1)$. After k^* samples, the applied control signal shifts and becomes $u_k = -x_k/c$. Hence the unstable first order test problem representation in discrete closed form is:

$$x_k = (a - b/c)^k x_0 \quad (52)$$

In terms of using stability requirement for the expression (52), the bounds on the c parameter can be obtained.

4.1.4 Saturation Boundaries for Convergent Dynamics

To guarantee stability the root of $(a - b/c)$ in (52) must be between 0 and 1 for a discrete first order system. Thus, the boundaries of the c cut off sample are:

$$0 < (a - b/c) < 1, \text{ which is } b/a < c < 1 \quad (53)$$

During the simulation this constraint on c should always be satisfied. As the control signal is bounded, the cut off sample c together with the initial state x_0 must be less than 1 to guarantee convergent dynamics. If c has a value around zero, then the system shows chaotic dynamics rather than convergent dynamics. On the other hand, if it converges to a value which is slightly less than its minimum, then initially it has oscillatory dynamics but finally it converges to upright. These are all illustrated by Figure 17. To obtain only convergent dynamics a new lower bound for c is defined as:

$$0 < b \leq c < 1 \quad (54)$$

where b is the lower limit of c which yields convergent terminal dynamics. The upper bound for c must be less than 1 to have a well-posed learning problem. The next section introduces infinite horizon value function and quadratic state-control signal based reward.

4.1.5 Value Function

The value function measures/estimates the long-term performance of the RL algorithm. Here the value function analysis is based on the dynamics of the balance test problem described in section 4.1, although, in any real application the optimal system dynamics are unknown. A quadratic state-control signal based instantaneous reward, which is the measure of the quality of the instant learning, is used for this analysis.

$$r(x_k, u_k) = qx_k^2 + \lambda u_k^2 \quad (55)$$

Here $q > 0$ and $\lambda > 0$ are positive definite weighting factors for the state and control signal respectively. Generally speaking having large q value means in order to minimize the reward $r(x_k, u_k)$, the state x_k must be smaller. The same argument is valid for λ as well. The exponentially weighted value function with zero terminal cost is given by:

$$V(x_k) = \sum_{t=0}^{\infty} \gamma^t r(x_{k+t}, u_{k+t}) \quad (56)$$

In general, the value function depends on the control signal and also the system parameters, but they are all dropped. It is important to note that the value function learning with a quadratic state-control signal based reward is performance based rather than stability based, which occurs in many RL applications. In the literature other kinds of rewards such as binary (for stability or minimum time) or absolute state based rewards are considered. However, if the reward is binary, then the closed form solution of the terminal value function expressed in section 4.2 will not have linear and quadratic parameters where both of them occur due to reward signal. On the other hand, with an absolute state based reward, quadratic terms in the closed form terminal value function representation will not appear as the value function learning analysis in chapter 3 shows. Section 4.2 analysis the learning properties of the value function in terms of using the introduced quadratic state based reward.

4.2 Closed Form Terminal Value Function Expression

To explicitly analyze the TD algorithms, particularly TD(0) and residual gradient learning algorithms, a closed form solution of the terminal value function for the test

problem is now derived. This expressed closed form solution will create a base for the parameter convergence analysis in this section and the simulation experiment performed in the next section.

Theorem 4.1: The closed form solution of the terminal value function for the balance test problem with the saturated piecewise linear control yielding convergent dynamics with integer k^* is given by:

$$\begin{aligned}
V(x_k) = & w_0^2 \left(\frac{q}{1-\gamma} + \frac{\lambda s(x_0)^2}{(1-\gamma)w_0^2} \right) - |x_0 - w_0| q \frac{2}{1-\gamma a} + (x_0 - w_0)^2 q \frac{1}{1-\gamma a^2} \\
& + |x_0 - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \left(w_{0,c}^{\frac{\ln(\gamma a)}{\ln(a)}} \frac{2\gamma q a}{1-\gamma a} - w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \frac{\gamma (q w_0^2 + \lambda s(x_0)^2)}{1-\gamma} - w_{0,c}^{\frac{\ln(\gamma a^2)}{\ln(a)}} \frac{\gamma q a^2}{1-\gamma a^2} \right) \\
& + x_{k^*+1}^2 |x_0 - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \left(\frac{q\gamma}{1-\gamma a(c)^2} + \frac{\lambda\gamma}{c^2 (1-\gamma a(c)^2)} \right)
\end{aligned}$$

where $w_0 = bs(x_0)/(a-1)$ and $w_{0,c} = (|w_0| - c)$.

Proof: Initially consider the closed form value function solution until the cut off sample k^* which has representation in terms of the plant dynamics as described by (51).

$$V' = \sum_{k=0}^{k^*} \gamma^k r(x_k, u_k) \quad (57)$$

Here $V' = V'(x_k, u_k)$ is the closed form terminal value function solution of the test problem until the sample k^* . Expanding the introduced quadratic state based reward given by (55) and substituting this reward in the value function representation in (57) gives:

$$V' = q \sum_{k=0}^{k^*} \gamma^k \left(w_0^2 + 2a^k w_0 (x_0 - w_0) + a^{2k} (x_0 - w_0)^2 \right) + \lambda \sum_{k=0}^{k^*} \gamma^k (-s(x_0))^2 \quad (58)$$

To simplify the closed form solution of the terminal value function denote $w_0(x_0 - w_0) = -|x_0 - w_0|$. Hence the first part of the solution is:

$$V(x_k) = w_0^2 \left(\frac{q}{1-\gamma} + \frac{\lambda s(x_0)^2}{(1-\gamma)w_0^2} \right) - |x_0 - w_0| q \frac{2}{1-\gamma a} + (x_0 - w_0)^2 q \frac{1}{1-\gamma a^2} \\ + |x_0 - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \left(w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \frac{2\gamma q a}{1-\gamma a} - w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \frac{\gamma (q w_0^2 + \lambda s(x_0)^2)}{1-\gamma} - w_{0,c}^{\frac{\ln(\gamma a^2)}{\ln(a)}} \frac{\gamma q a^2}{1-\gamma a^2} \right) \quad (59)$$

It is clear from (59) that the solution has a bias, linear, quadratic and higher order polynomials having representation in terms of the discount factor and plant dynamics. For the value function solution from $k^* + 1 \rightarrow \infty$, the system dynamics described by (52) will be used.

$$V'' = q x_{k^*+1}^2 \gamma^{k^*+1} \sum_{k=0}^{\infty} \gamma^k (a(c)^k)^2 + \lambda \frac{x_{k^*+1}^2}{c^2} \gamma^{k^*+1} \sum_{k=0}^{\infty} \gamma^k (a(c)^k)^2 \quad (60)$$

where $V'' = V''(x_k, u_k)$ is the closed form terminal value function solution for the sample interval of $[k^* + 1, \infty]$, x_{k^*+1} is the state just after the cut off sample and $a(c) = (a - b/c)$ is the system dynamics in the linear control region. The closed form solution for (60) yields:

$$V'' = x_{k^*+1}^2 |x_0 - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \left(\frac{q\gamma}{1-\gamma a(c)^2} + \frac{\lambda\gamma}{c^2(1-\gamma a(c)^2)} \right) \quad (61)$$

Combining V' given by (59) and V'' given by (61) gives terminal value function solution for the test problem which is the proof for theorem 4.1.

□

Step by step derivation of the terminal value function solution given by (61) is derived in Appendix B. The piecewise nature of the polynomial terms is due to the fact that the instantaneous reward is the absolute quadratic value of the state and control signal [78].

Remark 4.1: The value function solution has a constant term, linear term, quadratic term and two higher order terms, where the order depends on γ and a parameters. The constant, linear and quadratic terms are not surprising since the instantaneous

reward is a quadratic state-control signal based reward. These higher order parameters occur due to effect of the piecewise change in the control signal at $x_k = c$.

For theorem 4.1, it is assumed that k^* is an integer. Nevertheless, this is true if the chosen initial state exactly hits the cut off state c . The next section defines the effect of non-integer k^* on the closed form terminal value function expression.

4.2.1 Effects of Non-integer Cut off Sample k^*

Even though an integer cut off sample assumption is necessary for the closed form terminal value function derivation, in reality this is only achievable for a certain set of discrete initial states. To reveal the impact of non-integer k^* , real value function representation is compared with the closed form solution derived in section 4.2.

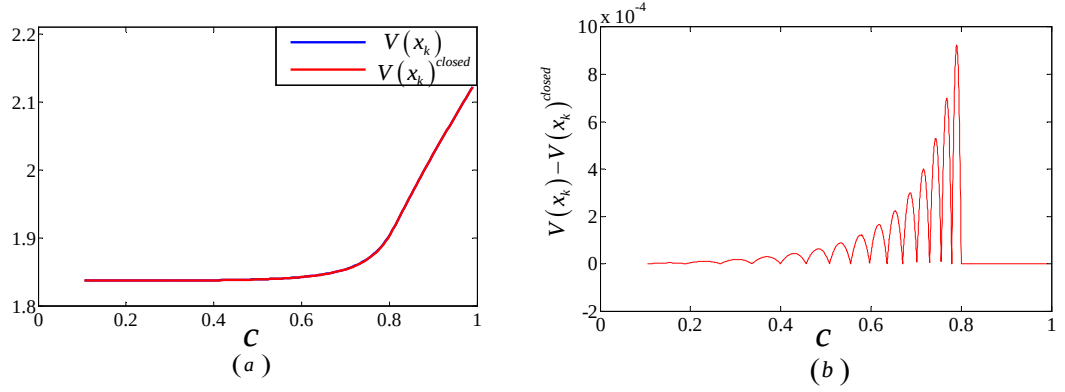


Figure 18: Comparison of the real and closed form value function solutions, a) k^* is non-integer, b) error when k^* is non-integer.

Here $V(x_k)^{closed}$ is the closed form value function solution. Simulation parameters are the same as defined in section 4.1.2. The initial state is $x_0 = 0.8$, the discount factor is $\gamma = 0.7$ and when $c < x_0$ the effect of non-integer cut off sample appears. The error occurring due to non-integer k^* sample disappears with smaller c as shown in the Figure 18. This is because of the discount factor which discounts the initial states heavily and then exponentially decays to zero.

4.2.2 Optimal Polynomial Basis and Parameter Vector

The optimal parameters of the terminal value function are given by:

$$w^* = \left[\left(\frac{q}{1-\gamma} + \frac{\lambda s(x_0)^2}{(1-\gamma)w_0^2} \right) - \frac{2q}{1-\gamma a} - \frac{q}{1-\gamma a^2} \quad w_4^* \quad w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \left(\frac{q\gamma}{1-\gamma a(c)^2} + \frac{\lambda\gamma}{c^2(1-\gamma a(c)^2)} \right) \right] \quad (62)$$

$$\text{where } w_4^* = \left(w_{0,c}^{\frac{\ln(\gamma a)}{\ln(a)}} \frac{2q\gamma a}{1-\gamma a} - w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \frac{\gamma(qw_0^2 + \lambda s(x_0)^2)}{1-\gamma} - w_{0,c}^{\frac{\ln(\gamma a^2)}{\ln(a)}} \frac{\gamma q a^2}{1-\gamma a^2} \right) \text{ is the higher}$$

parameter occurring at the sample k^* . The k^{th} estimate of the parameter vector is represented as:

$$w_k = [w_{1,k} \quad w_{2,k} \quad w_{3,k} \quad w_{4,k} \quad w_{5,k}] \quad (63)$$

As the bias parameter just depends on the discount factor, it is not necessary to learn it.

Remark 4.2: It should be emphasized here that the terminal value function has five unknown parameters to estimate even though the dynamic system in the exemplar balance problem has just two parameters. The optimal polynomial basis of the terminal value function is given by:

$$\phi(x_k, w_k) = [\underbrace{w_0^2}_{\text{Basis for Bias Parameter}} \quad \underbrace{|x_k - w_0|}_{\text{Basis for Linear Parameter}} \quad \underbrace{|x_k - w_0|^2}_{\text{Basis for Quadratic Parameter}} \quad \dots \quad \underbrace{|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}}}_{\text{Basis for Higher Order Polynomial Occuring at } k^*} \quad \underbrace{x_{k^*+1}^2 |x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}}}_{\text{Basis for Higher Order Polynomial Occuring at } k^*+1}] \quad (64)$$

Remark 4.3: Both higher order parameters occurring at the sample k^* and k^*+1 are due to the effect of the change in the character of the control signal where it is necessary to estimate the power as it depends on the plant parameter a together the learning parameter γ .

4.3 Trajectory Null Space of the TD Error

The TD error plays a crucial role in the value function learning as its parameter update process depends on the TD error. Hence, if the TD error is incapable of representing information about a particular parameter, then optimal parameter learning process will be hindered. In this section, it is shown that the TD error, during the state trajectory up to k^*+1 , has a null space associated with the differenced bases of the higher order

parameters. Moreover, it is also expressed that this has a different effect on the TD(0) and residual gradient learning algorithms but both of the algorithms are unable to learn the higher order parameters until $k^* + 1$.

Theorem 4.2: TD error has a non-zero null space up to $k^* + 1$ sample along the optimal trajectory as the differenced bases associated with the higher order parameters are:

$$\phi(x_k, w_{4:5,k}) - \gamma \phi(x_{k+1}, w_{4:5,k}) = 0$$

where $\phi(x_k, w_{4:5,k})$ are the bases for the higher order parameters.

Intuitive Proof: Assume that the higher order parameters are $w_{4:5,k} \neq 0$ and the corresponding differenced bases are $\phi(x_k, w_{4:5,k}) - \gamma \phi(x_{k+1}, w_{4:5,k}) \neq 0$. Then, the estimated reward which is given by:

$$\hat{r}(x_k, w_k) = (\phi(x_k, w_k) - \gamma \phi(x_{k+1}, w_k)) w_k^T$$

spans in the parameter space with the bias $w_{1,k}$, linear $w_{2,k}$, quadratic $w_{3,k}$, higher order $w_{4,k}$ and $w_{5,k}$ parameters.

$$\hat{r}(x_k, w_k) = \langle w_{1,k}, w_{2,k}, w_{3,k}, w_{4,k}, w_{5,k} \rangle$$

As discussed in section 4.2, the quadratic instantaneous reward span in the parameter space with the bias, linear and quadratic parameters.

$$r(x_k) = \langle w_{1,k}, w_{2,k}, w_{3,k} \rangle$$

In this case, the TD error which has representation as:

$$\delta_k = r(x_k) - \hat{r}(x_k, w_k)$$

is non-zero. Thus, the differenced higher order bases must be:

$$\phi(x_k, w_{4:5,k}) - \gamma \phi(x_{k+1}, w_{4:5,k}) = 0$$

in order to yield zero TD error [78]. This shows that the higher order parameters cannot be learned until the $k^* + 1$ sample is reached.

Proof: The optimal polynomial basis function given by (64) is used for the null space analysis of the differenced basis function. Here, w_0 and the power parameters are assumed to be known and $\Phi(x_k, w_{4,k})$ represents the differenced basis for the higher order parameter generated at k^* .

$$\begin{aligned}\Phi(x_k, w_{4,k}) &= \left(\phi(x_k, w_{4,k}) - \gamma \phi(x_{k+1}, w_{4,k}) \right) \\ &= \left(|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} - \gamma |x_{k+1} - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \right) \\ &= \left(|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} - \gamma |ax_k - bs(x_k) - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \right)\end{aligned}\quad (65)$$

Since $-bs(x_k) - w_0 = -bs(x_k) - bs(x_k)/(a-1)$ and this is equal to $-abs(x_k)/(a-1)$ which is basically $-aw_0$, it becomes:

$$\begin{aligned}\Phi(x_k, w_{4,k}) &= \left(|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} - \gamma |ax_k - aw_0|^{\frac{\ln(\gamma)}{\ln(a)}} \right) \\ &= \left(|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \left(1 - \gamma \left(e^{\ln(a)} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \right) \right)\end{aligned}\quad (66)$$

Finally, $\ln(a)$ in (66) disappears and $e^{-\ln(\gamma)}$ becomes $1/\gamma$. Hence, the equation (66) is:

$$\Phi(x_k, w_{4,k}) = \left(|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \left(1 - \gamma \frac{1}{\gamma} \right) = 0 \right) \quad (67)$$

The fourth column which is the basis associated with the higher order parameter is zero.

To prove that the differenced basis associated with the higher order parameter occurring at $k^* + 1$ is zero, consider an initial state which is larger than the cut off state. Since x_{k^*+1} is a constant for $x_k \geq c$ and as the differenced higher order basis is zero up to the cut off sample, this higher order basis will be zero as well.

$$\begin{aligned}
\Phi(x_k, w_{5,k}) &= \left(x_{k^*+1}^2 |x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} - \gamma x_{k^*+1}^2 |ax_k - aw_0|^{\frac{\ln(\gamma)}{\ln(a)}} \right) \\
&= \left(|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} x_{k^*+1}^2 \left(1 - \gamma \left(e^{\ln(a)} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \right) \right) = 0
\end{aligned} \tag{68}$$

Therefore, these higher order parameters will not be learnt until the sample $k^* + 1$ since the corresponding differenced bases are both zero.

□

Henceforth, both the expected and instantaneous rewards span in the parameter space without carrying any information about the higher order polynomials up to $k^* + 1$ sample. Based on this theorem two important corollaries are now given to explain the impact of this non-zero null space on TD(0) and residual gradient learning algorithms.

4.3.1 Trajectory Null Space Effect on Residual Gradient Learning

It is clear that both the residual gradient and TD(0) algorithms explicitly depend on the TD error. Therefore, the instantaneous estimation of the higher order parameters will be poor as the TD error is independent of these parameters until the $k^* + 1$ sample. Corollary 4.1 addresses this case for the residual gradient algorithm.

Corollary 4.1: The higher order parameter update is zero for the residual gradient learning algorithm until the $k^* + 1$ sample is reached.

Proof: Consider the residual gradient learning parameter update rule introduced in 2.4.4, where the differenced basis function is part of the update:

$$\Delta w_k = \eta \left(r(x_k) - (\phi(x_k, w_k) - \gamma \phi(x_{k+1}, w_k)) w_k^T \right) (\phi(x_k, w_k) - \gamma \phi(x_{k+1}, w_{k+1})) \tag{69}$$

Theorem 4.2 shows that the differenced basis associated with the TD error has a non-zero null space. Therefore, the TD error does not carry any information about the higher order parameters. Moreover, the differenced basis occurring due to gradient of the differenced value function has no information about these parameters. Furthermore, the corresponding parameters update $\Delta w_{4:5,k} = 0$ for the residual gradient learning.

□

The simulation result presented in section 4.5 justifies that there is no update for these particular parameters during the state trajectory. As theorem 4.2 and corollary 4.1 stated that the TD error does not contain any information about the higher order parameters, $w_{4:5,k}$, as also occurs with the TD(0) algorithm.

4.3.2 Impact of the Trajectory Null Space on TD(0) Learning

The non-zero null space has a different effect on the higher order parameters learning with TD(0) due to difference in the formulation as corollary 4.2 expresses.

Corollary 4.2: The higher order parameter update is non-zero for the residual gradient learning algorithm up to $k^* + 1$ sample.

Proof: The TD error and hence the TD(0) learning algorithm described in 2.4.3 is independent of the higher order parameters $w_{4:5,k}$ during the state trajectory until the $k^* + 1$ sample. The higher order parameters, $w_{4:5,k}$, do not affect the TD(0) because it is multiplied by the zero, differenced basis function. However, because of the formulation of the TD(0) update, these parameters are still updated because the update rule is based on the basis function vector at time k , not the difference:

$$\Delta w_k = \eta \left(r(x_k) - (\phi(x_k, w_k) - \gamma \phi(x_{k+1}, w_k)) w_k^T \right) \phi(x_k) \quad (70)$$

While this causes non-zero weight updates, $\Delta w_{4:5,k} \neq 0$, this should be regarded as parameter drift during the trajectory because the errors are independent of these parameters.

□

Therefore, both the residual gradient and TD(0) learning algorithms are unable to learn these higher order parameters until the sample $k^* + 1$ because of the non-zero null space. The simulation results justify the theorem and these two corollaries.

4.4 Badly Conditioned Learning Problem

In this section it is shown that the learning problem can become badly conditioned depending on the value of the discount factor and saturation threshold.

Theorem 4.3: The learning problem is badly conditioned as the discount factor and the saturation threshold approach $1/a$ and zero, respectively.

Proof: For the sum to converge, the value function must have $\gamma a < 1$ or $\gamma < 1/a$. In the value function expression, the linear parameter and the higher order parameter occurring at the k^* sample approach ∞ as $\gamma \rightarrow 1/a$. Similarly, the corresponding higher order basis becomes:

$$|x_k - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \rightarrow |x_k - w_0|$$

which is the linear basis for the linear parameter and this leads to singular condition problem.

As the saturation threshold c gets small, then x_{k^*+1} , which is the state just after the saturation threshold, will be smaller. Furthermore, the higher order basis associated with the higher order parameter occurring at $k^* + 1$ approaches zero as $c \rightarrow 0$.

$$x_{k^*+1}^2 |x_0 - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \rightarrow 0$$

Therefore, the amount of information in the higher order basis will reduce as the saturation threshold gets close to zero and this causes badly/poor conditioned learning for this higher order parameter [78].

□

Thus, as the cut off sample and the discount factor approach zero and $1/a$, respectively, then the condition number of the differenced basis function increases and the learning problem becomes badly conditioned. As a result, the learning process will be sluggish and poor rate of convergence will occur.

4.5 Simulation

In this section, the parameter convergence and rate of convergence associated with the terminal value function for the exemplar balance problem with the saturated piecewise linear control is simulated. For simulation Matlab software is used. The relationship to the traditional LMS algorithm is highlighted by illustrating how the data is projected onto the differenced basis function vector in parameter space. The null space associated with the TD error up to the cut off sample is illustrated and the rate of convergence of the residual gradient and the TD(0) learning algorithms are compared.

4.5.1 Simulation Parameters and Algorithm

The discrete time exemplar balance control problem dynamics for the saturated part of the control signal is given by:

$$x_{k+1} = 1.1051x_k - 0.1051s(x_k) \quad (71)$$

This can be attained from the continuous time system by a sample time of 0.1s with a unity time constant. Hence, the test problem parameters are $a = 1.1051$ and $b = 0.1051$. Without loss of generality, the initial state is assumed to be positive and the corresponding optimal control action is piecewise linear where the cut off parameter c is equal to the b parameter; hence chaotic dynamics do not appear. Thus, the system dynamics after the k^* sample for this specific test problem becomes:

$$x_{k+1} = (a - b/b)x_k, \text{ which is } x_{k+1} = bx_k \quad (72)$$

The simulation consists of 25000 batches and each experiment consists of 75 samples which takes the initial state $x_0 = 0.9940$ to the upright position. While the length of the simulation seems large, this is required due to badly conditioned learning as shown later.

The representation of the terminal value function uses the optimal polynomial basis defined by Theorem 4.1 and it assumes that w_0 and the power parameters are known but the rest of the parameters will be learnt. While this is an unreasonable assumption in practice, it simplifies the analysis of parameter convergence performed in this chapter. The discount factor is $\gamma = 0.7$, the gain of the system is 1; so $w_0 = 1$. Hence

the terminal value function solution with this discount factor and a, b, c system parameters:

$$V(x_k) = 3.33 - 8.83|x_k - 1| + 6.89|x_k - 1|^2 - 1.39|x_k - 1|^{3.56} + 1.04x_{k+1}^2|x_k - 1|^{3.56} \quad (73)$$

Now the task is to estimate these parameters using the defined optimal polynomial basis.

$$\phi(x_k) = \begin{bmatrix} 1 & |x_k - 1| & |x_k - 1|^2 & |x_k - 1|^{\frac{\ln(\gamma)}{\ln(a)}} & x_{k+1}^2 |x_k - 1|^{\frac{\ln(\gamma)}{\ln(a)}} \end{bmatrix} \quad (74)$$

The simulation starts with a zero initial parameter vector.

The maximum value of the basis $\|\phi(x_k) - \gamma\phi(x_{k+1})\|_2^2$ is 0.36 for RG (where RG stands for the residual gradient) while the maximum value of the basis $(\phi(x_k) - \gamma\phi(x_{k+1}))^T \phi(x_k)$ is 1.2 for the TD(0) algorithm. Furthermore, it is clear that $\|\phi(x_k) - \gamma\phi(x_{k+1})\|_2^2 \ll (\phi(x_k) - \gamma\phi(x_{k+1}))^T \phi(x_k)$ so to compare the convergence rates, the learning rates should be chosen to reflect this. Learning stability is assured if the learning rates lie in the intervals $0 < \eta^{RG} < 5.55$ and $0 < \eta^{TD(0)} < 1.66$. Thus, in order to compare the rates of convergence of the different schemes, the learning rates should be selected in the same manner, not necessarily the same value. The residual gradient learning rate should be significantly larger than for the TD(0), and in this simulation the learning rates are set to be $\eta^{RG} = 0.72$ and $\eta^{TD(0)} = 0.22$ which are 13 percent of their maximum values. Obviously, when there exists significant noise/modelling error, the learning rates will have to be considerably smaller.

4.5.2 Bias and Linear Parameter Convergence

Initially, the (projected) parameter convergence of the bias, w_1 , and the linear, w_2 , parameters will be analyzed for both the residual gradient and the TD(0) algorithms.

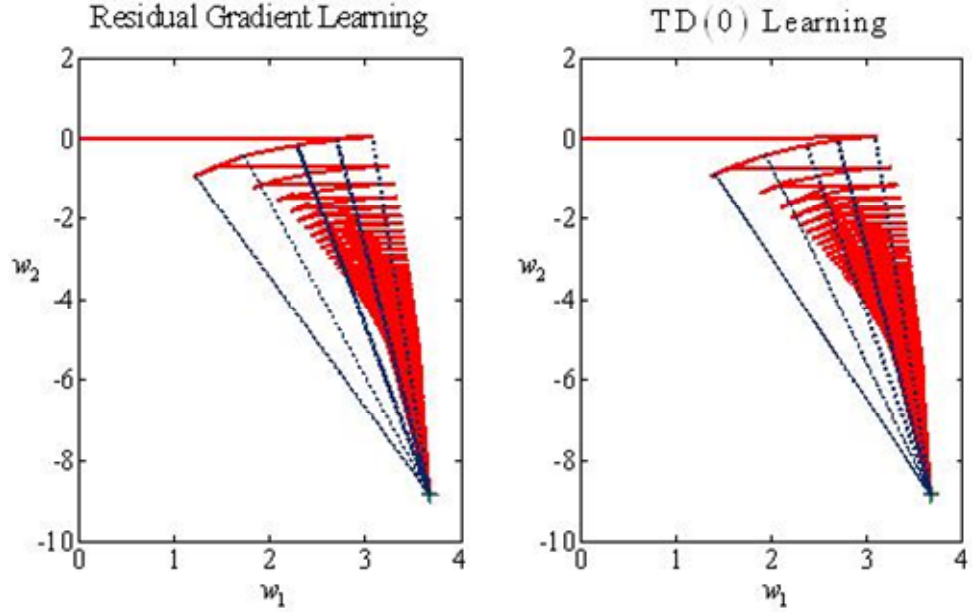


Figure 19: Convergence for the bias w_1 and linear w_2 parameters of the residual gradient and TD(0) learning.

The solid red line shows the parameter trajectories and dotted blue line represents the solution lines associated with each training datum (corresponding to the differenced basis functions). As shown in Figure 19, the parameter trajectory has a repeated zigzag pattern in the bias and linear parameter learning in which each pattern represents a batch consists of 75 samples. Both of the parameters have initial value of zero and initially the bias term associated with the differenced basis function vector is near zero and the parameters are projected horizontally towards the solution hyperplane (set of all parameter values that satisfy that datum) whereas the linear basis is almost zero. Therefore, the bias parameter is the first parameter which starts learning. As the system dynamics move towards the upright equilibrium, the linear term in the differenced basis function vector becomes significant and the solution hyperplane rotates in an anti-clockwise direction, until it lies at an angle of about 45° with the upright. During the experiment, the parameter estimates track the rotating solution hyperplane. This is similar for both the residual gradient and the TD(0) algorithms [78, 79].

It is important to note that at the end of each batch, the final parameters for the residual gradient and TD(0) learning do not reach the same solution line due to the fact that the TD error is not totally independent of the higher order parameters

learning. Although the residual gradient and TD(0) learning algorithms have similar convergence results for these parameters, this difference is more significant for the higher order parameter learning as discussed next.

4.5.3 Quadratic and Higher Order Parameter Convergence

Parameter convergence for the quadratic parameter w_3 and higher order parameter w_4 are partially given for the residual gradient and TD(0) learning.

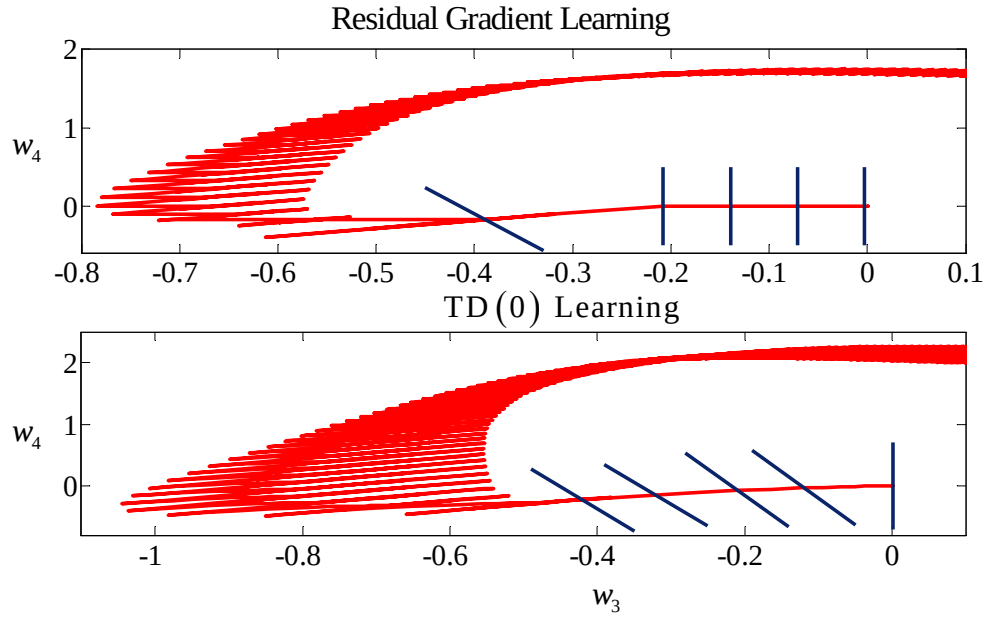


Figure 20: w_3 and w_4 parameter convergence of residual gradient and TD(0) learning.

The red curves represent parameter trajectories and dotted blue lines are solution trajectories shown just for one batch. Figure 20 shows the parameter trajectory associated with estimating the quadratic, w_3 , and higher order, w_4 , basis function parameters. The convergence of w_4 differs from the bias, linear and quadratic parameters since the corresponding residual gradient update is zero up to the cut off sample as stated by theorems 4.2 and 4.3. However, after the cut off sample this higher order parameter is learned as shown by Figure 20. For the terminal point, where the pendulum remains stationary in the upright position, the solution hyperplane is rotated; the parameter vector is projected onto it. Then the experiment is repeated and the parameter update trajectory is repeated. For TD(0), w_4 actually has a non-zero update when the pendulum is rotating towards the upright. However, it should be observed that it is drifting away from the true value. The only correction for

which the TD error actually depends on w_4 is after the cut off sample k^* . Together with this higher order parameter, the other higher order parameter has no information in its own basis. Therefore, it has same the learning characteristics as w_4 as shown next.

4.5.4 Higher Order Parameter Convergence

Parameter convergence for the two higher order parameters w_4 and w_5 are partially shown for the residual gradient and TD(0) learning.

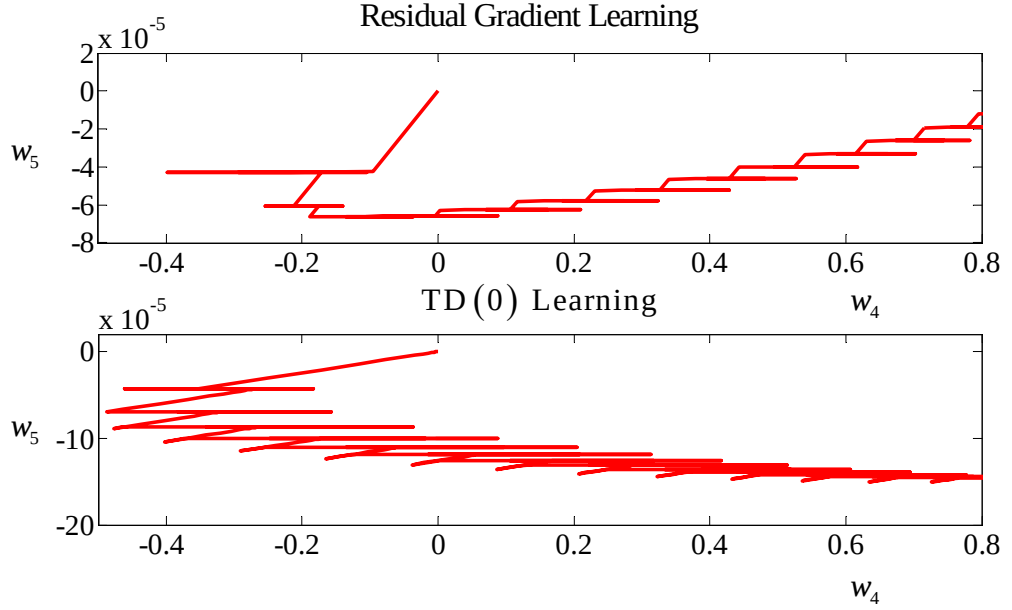


Figure 21: Higher order parameters convergence for the residual gradient and TD(0) learning.

As stated by Theorem 4.2, neither of the higher order parameters can be learnt since the corresponding bases do not carry any information about these parameters up to the saturation threshold. Based on this theorem, corollaries 4.1 and 4.2 expressed that the parameter update is zero for RG learning whereas the parameter update is non-zero for TD(0) learning but the parameters are just drifting away as shown in Figure 21. It is noticeable that even though both of the higher order parameters start learning just after the cut off sample, the higher order parameter associated with the linear part of the control signal has smaller parameter update compare to the other higher order parameter. As the Theorem 4.3 stated, this higher order parameter learning is badly conditioned if the cut off sample $c \rightarrow 0$. In this case, in order to learn the correct

higher order parameter either a significantly large learning rate or longer simulation time must be provided. Since sufficient time for this badly conditioned learning cannot be given due to limited computer capacity, the error in parameter estimation process will not be zero as addressed next.

4.6 Rate of Convergence

Determining the speed at which the parameters approach their correct values is important as it provides insights about the performance of the learning algorithms. Parameter convergence (norm of the parameter error) is shown in Figure 22 for both the residual gradient and TD(0) learning algorithms.

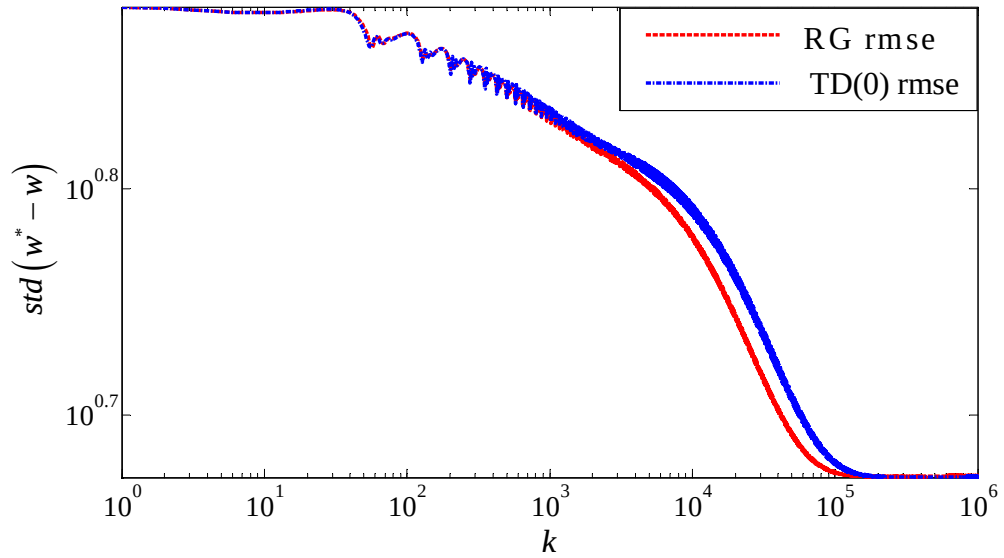


Figure 22: Root Mean Square (RMS) parameter error comparison of the residual gradient and TD(0) learning.

Here k represents the samples and $std(w^* - w)$ is the RMS error between the correct and estimated parameters. It is clear from Figure 22 that the residual gradient learning converges at a faster rate asymptotically. As discussed above, the bias, linear and quadratic parameters of the residual gradient learning algorithm are learnt during the whole batch whereas the parameters with the higher order are learnt from just after the cut off sample to the samples. As a result of this property, there is a sharp reduction in the RMS error of the value function parameters after the cut off sample in each batch (difficult to see on the graph as all the 25000 batches are shown). For TD(0) however, even though all parameters are updated at each time sample, convergence is slower. It

was noted that the higher order parameter seemed to be drifting away within each batch and the higher order parameters are updated in the correct direction just after the cut of sample. Henceforth, the RMS error reduction is significant after the cut off sample. It is also noticeable that the RMS error is not zero due to badly conditioned higher order parameter learning which requires either infinitely long time or larger learning rate to converge to its correct value for this defined cut off value.

To quantify the rate of convergence, condition number and time constant of the corresponding bases can be considered. The condition number of the differenced bases are $2.92e+6$ and $2.99e+6$ for the residual gradient and TD(0) algorithms respectively. Thus, it is clear that the TD(0) algorithm has slightly worse conditioned basis and this leads to slower convergence. Same insight can be obtained from the time constants of the learning algorithms as well. The time constants are $-1/\log(1-\eta^{RG}\sigma_{\min}^{RG}) = 7.24e+3$ for the residual gradient and $-1/\log(1-\eta^{TD(0)}\sigma_{\min}^{TD(0)}) = 7.27e+3$ for the TD(0) algorithms respectively, where $\sigma_{\min}^{RG}$ and $\sigma_{\min}^{TD(0)}$ represent the minimum eigenvalues of the bases associated with the residual gradient and TD(0) algorithms. Since a smaller time constant yields faster convergence, the residual gradient algorithm converges slightly faster than TD(0) algorithm [78, 79].

4.7 Summary

This chapter introduced a simplified balance test problem with saturated piecewise linear control that can be used as a test bed to generically evaluate the value function's parameter convergence/rate of convergence by using TD learning. To express learning results explicitly, the optimal value function and its corresponding polynomial basis were calculated. It is proved that the higher order bases introduce a trajectory null space up to k^*+1 in TD learning; hence the corresponding convergence process is slow as there is no information available about these parameters. Moreover, it is expressed that badly conditioned learning occurs for the higher order parameter associated with the linear part of the control signal if the cut off value approaches zero. In addition, it is noted that the learning problem is badly conditioned as the discount factor γ approaches $1/a$. A simulation experiment is performed in order to

show the convergence and rate of convergence results for the residual gradient and TD(0) learning algorithms. The simulation results validate the theorems stated in section 4.3 and also show that the residual gradient algorithm is converging faster than the TD(0) algorithm for this test problem.

5 RL Simulations and Learning Analysis for Higher Order System

In chapters 3 and 4, closed form terminal value function expressions are derived for two test problems. This approach reveals important information about the value function's learning properties; parameter convergence, rate of convergence and the role of the control signal in learning. As the optimal polynomial bases are assumed to be available, effects of value function and control policy modeling errors have not been considered or discussed. In addition, it was assumed that the optimal control policy was known; hence the exploration approaches to learn the optimal control policy have not been taken into account. However, in real life RL applications it is not possible to assume that optimal polynomial bases for the value function and control policies are available, as the dynamics of the systems are unknown and typically the plant is non-linear. In this chapter, RL is applied to a higher order plant and by using the gained insights from the previous chapters, the value function and control policy learning results are discussed.

5.1 RL with Function Approximation and Exploration Noise

Application of the RL algorithms to higher order systems requires use of function approximation to represent the value function and control policy. Hence, the effects of the modelling error must be explicitly considered and also approaches for reducing the dimensionality of the approximation algorithm. Moreover, exploration noise is an important tool in RL to learn optimal control policies. This section discusses the necessity and role of the function approximation and exploration noise in RL.

5.1.1 RL with Function Approximation

In practical RL approaches, it is not possible to derive the exact value function or control policy representations, since the optimal basis derivation performed in the previous sections cannot be applied in practice. Therefore, function approximation techniques must be considered. RL with function approximation brings a number of questions such as:

- Does the RL algorithm converge when modeling error exists?

- Is the learned solution with the RL algorithm satisfying the control targets?

Obtaining generic answers for these questions is not possible since the closed form terminal value function solution and optimal control policy are both unavailable [55]. However, in terms of using the gained insights with the test problem analysis learned approximate solutions can be analyzed. A further key property in RL is exploration noise discussed next.

5.1.2 RL with Exploration Noise

The previous value function learning analyses performed by considering a test problem do not have exploration and exploitation properties since it was assumed that the optimal control policies were known and the goals were to explicitly analyze the value function parameter convergence and rate of convergence. However, as expressed in chapters 1 and 2, RL algorithms learn from experience by performing a trial and error process to obtain optimal control policies. Value function and controller weights are updated in direction of the learning outcome which is success or failure. Therefore, in order to obtain optimal value function and control policies, RL algorithms must interact with the environment. Basically, RL algorithm must take random actions to learn optimal control policies which satisfy the learning goals. Control policy merged with an exploration term prevents the learning algorithm from getting trapped in local minima; hence the role of exploration is to search alternative paths to attain best policies by taking control actions which deviate from previous ones [45]. If the RL algorithm sees the same states and takes the same control actions, then the learning algorithm will be unable to improve its experience even though the learned control policy is non-optimal. Once the best approximated optimal control policy is learned and the value function is maximized, then RL algorithm switches to exploitation phase where experience just gained from the exploration process is used.

5.2 Simulation Parameters and Setting for RL Analysis

This section presents simulation setting and parameters that will be used for the optimal and learned value function/control policy analysis. For simulations Matlab software is used.

5.2.1 Simulation Setting

Discrete time inverted pendulum model is used to analyze the learned control policies and value functions.

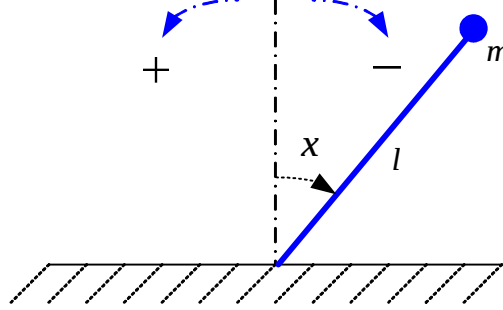


Figure 23: Inverted pendulum model.

The dynamics of the pendulum shown in Figure 23 is given by $ml^2\ddot{x} = mgl \sin(x) + u$, where mass and length of the pendulum are $m = 1$ kilogram (kg) and $l = 1$ meter (m), respectively and the gravitational acceleration is $g = 9.81 \text{ m/s}^2$. The linear discrete time state space representation of the pendulum is $x_{k+1} = Ax_k + Bu_k$, where $A = [1.049 \ 0.101; 0.996 \ 1.049]$ and $B = [0.005 \ 0.101]^T$ are the discrete parameters with a sample time 0.1 s. The initial position and velocity states of the pendulum are uniformly sampled between -0.3 and 0.3, -1 and 1, respectively for the optimal value function and control policy analysis. The maximum absolute value of the control signal is $U_{\max} = 7 \text{ N.m}$, where N represents Newton, which is large enough to ensure that the pendulum can converge to the upright from these initial states for the optimal solutions. The absolute value of the control signal is unity for the learned solutions and this is able to take the pendulum to the upright from the initial states inside the stability region.

5.2.2 Simulation Parameters

For this simulation a binary reward signal $r(x_k) = 1$ for success, $r(x_k) = 0$ for failure and a quadratic reward signal $r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k$ are considered. Here, $Q = [1 \ 0; 0 \ 1]$ and $R = 0.5$ are positive definite weighting matrices for the states and control signal of the pendulum, respectively. For the optimal value

function/control policy solution experiments, weighting factor for the control signal is adopted as $R/U_{\max}^2$ in order to have similar insights with the learned solutions. The instantaneous rewards are discounted by the discount factor $\gamma = 0.8$. Therefore, the value function discount factor is able to discount around for 20 future reward samples and this is large enough to see long term behaviour of the value function. A linear in the parameters RBF approximation scheme is used. Basis functions are defined as $\phi_k(x) = a_k(x) / \sum_{l=1}^K a_l(x)$ where $a_k(x) = e^{-\|(1/s_k^T)(x - c_k)\|^2}$, where s_k and c_k are the size and centre of the k^{th} basis function. There are ten uniformly sampled basis functions on each axis for the position (x_p) and velocity (x_v), where $|x_p| \leq 0.3$ and $|x_v| \leq 1$. Therefore, there are 100 multivariable basis functions. The size of the basis functions is $s_k = \left[(x_r(k+1) - x_r(k))/1.7 \quad (x_p(k+1) - x_p(k))/1.7 \right]$ and the centre of the basis functions is $c_k = \left[x_r(k) \quad x_v(k) \right]$. Basically, the size s_k of the RBF determines the width or radius of the bell-shape and the centre c_k of the RBF is distance from the input states. There also exists random exploration noise which is zero mean Gaussian and its standard deviation is given by $\sigma = \sigma_0 \left(\min \left(1, \max \left(0, V_{\max} - \hat{V}(x_k, w_k) \right) \right) \right)$, where $\sigma_0 = 0.5$.

Three different RL simulations are performed to analyze the RL properties. In the first simulation, the RL algorithm with a binary (success/failure) reward learns bang-bang control solution to control the pendulum towards the upright. In the second simulation, the instantaneous reward is a quadratic function of the state/control signal (LQR) and the aim is to learn the bang-bang control solution; hence minimize the energy of the state. Finally for the last simulation, the instantaneous reward is again quadratic in state/control signal and the control signal is saturated piecewise. This allows a more robust control policy to be learned.

5.3 RL Analysis with Binary Reward and Bang-Bang Control

In this section, RL is applied to the discrete pendulum model introduced in section 5.2. The instantaneous reward is binary (success/failure) and the RL algorithm attempts to learn a bang-bang control policy which keeps the pendulum in the stability

region. The bang-bang control solution and its modified version which imitates stability based control are briefly given. The value function is also presented in order to compare with the learnt ones.

5.3.1 Modified Bang-Bang Control Solution

Time optimal control law is reviewed in Appendix C. The control law is basically minimum time optimal control and its form is:

$$u_k = \begin{cases} -1 & \text{if } x_2(k+1) > (x_1(k+1)/|x_1(k+1)|)(1-(1-|x_1(k+1)|))^{-1} \\ 1 & \text{if } x_2(k+1) \leq (x_1(k+1)/|x_1(k+1)|)(1-(1-|x_1(k+1)|))^{-1} \end{cases} \quad (75)$$

where x_1 and x_2 are position and velocity states of the pendulum in modal form. Since the RL solution with success/failure based instantaneous reward is stability based, in order to imitate this type of control, Bernoulli distribution is combined with the (75). Random Gaussian noise n is generated and the control law is modified as:

$$u_{k^*} = \begin{cases} u_k & \text{if } n(k) < 0.8 \\ -u_k & \text{otherwise} \end{cases} \quad (76)$$

Therefore, randomness is introduced for the time optimal control law given by (75). The figure below illustrates the state trajectory of the pendulum with the control law expressed by (75) and (76).

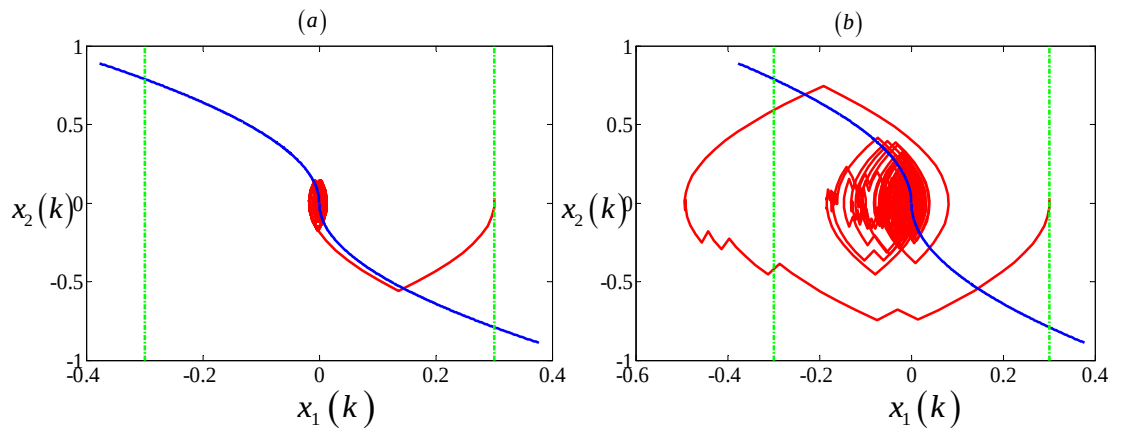


Figure 24: State trajectory; a) without noise, b) with noise.

The dotted green lines are located at $|x_1(k)| = 0.3$, the blue curve is the control decision curve and the red curve is the state trajectory with initial 0.3 and 0.03

position and velocity states, respectively where control law is as defined by the equation (76). As can be seen from Figure 24a), the bang-bang control law without uncertainty in control takes the pendulum to the origin. Due to discrete sampling the state trajectory does not exactly switch on the control decision curve and since these initial states do not yield zero terminal states, chaotic dynamics occur around the upright. With the added uncertainty, the control law takes random actions depending on the value of the generated random Gaussian data. As a result of this random control action, the state trajectory moves outside the green lines. In this case, it is accepted that the control algorithm has failed. Based on this insight, a stability region is defined next.

5.3.2 Stability Region and Control Surface

A stability region is described in order to analyze the impact of the non-zero trajectory null space occurring during failures. It is expressed in terms of the position state of the pendulum. If assigned initial states for the pendulum with the control law (76) yields position state less than 0.3 radians, then it is accepted that these initial states are inside the stability region.

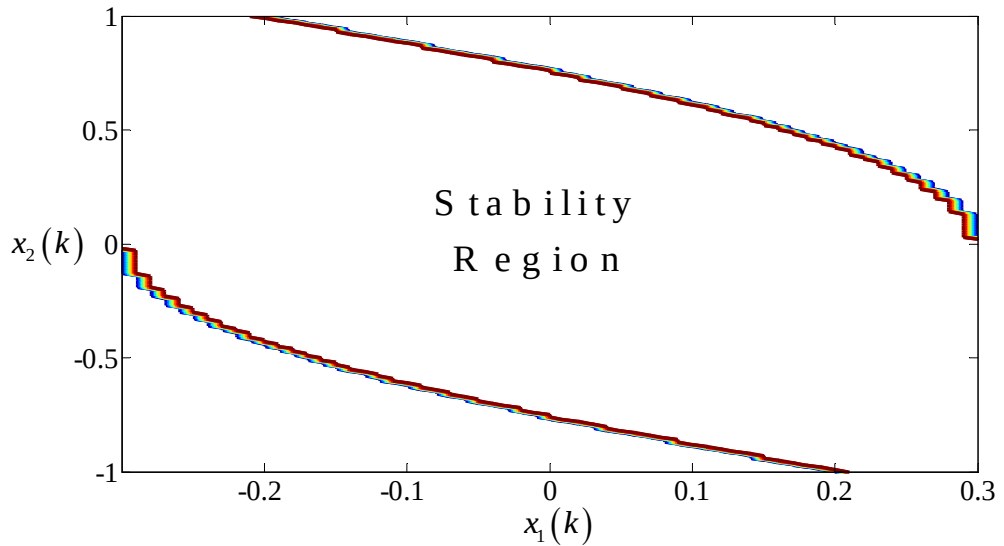


Figure 25: Defined stability region.

It is important to note that this stability region shown in Figure 25 is obtained without added uncertainty in the control action. This defined stability region provides basis for the value function analysis discussed in section 5.3.4.

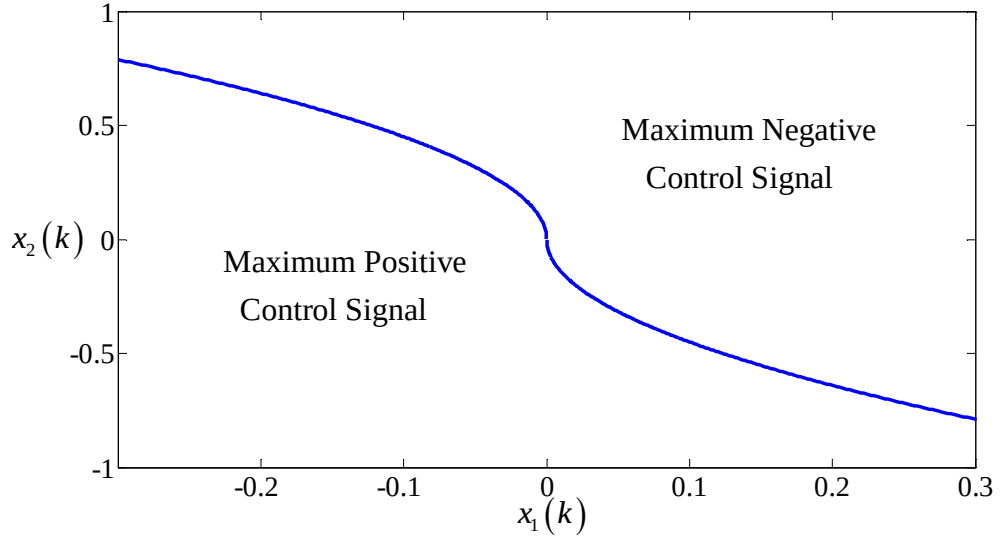


Figure 26: Bang-Bang control decision curve.

There are basically two regions in Figure 26 divided by the decision curve corresponding to the maximum/minimum control signals generated by the control law (76). The next section presents learned bang-bang control solution when the instantaneous reward is binary.

5.3.3 Learned Bang-Bang Control Policy

RL algorithm with a success/failure based instantaneous reward is able to learn the bang-bang control policy to drive states of the pendulum to origin.

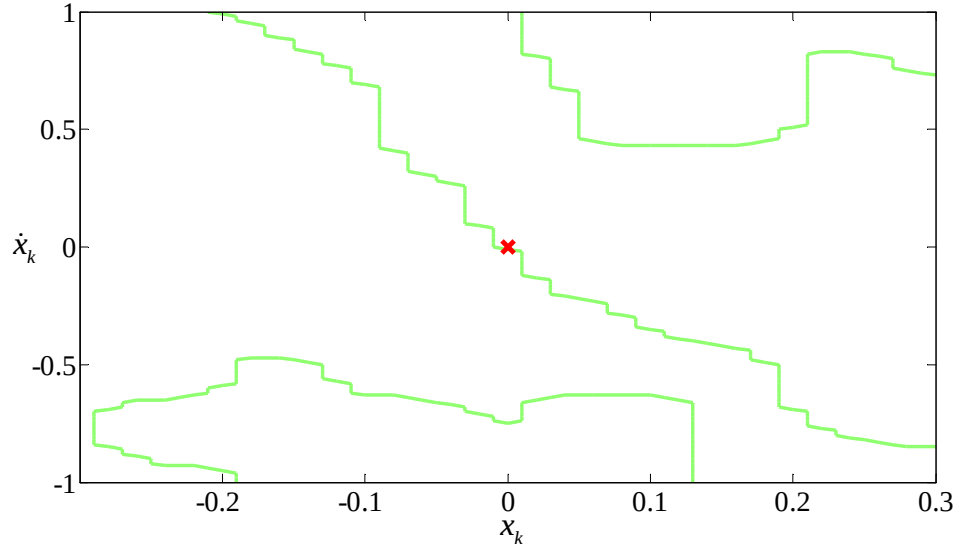


Figure 27: Learned bang-bang control policy with binary reward.

The red cross on the learned decision curve is located at the point where the position and velocity states are zero. The learned decision curve basically divides the state space into two and generates ± 1 unity control signal based on the learned control policy. As can be seen from the Figure 27, the decision curve passes through the origin. This shows that the pendulum states bang around the upright position due to generated bang-bang control signal. There are more curves around the decision curve which might occur because of the initial wrong learning or during failure of the RL algorithm. It is important to note that since RL with binary reward is stability based, the learned control policy $u_k = \text{sign}(\phi(x_k)^T w^a + w_0)$ might have non-zero w_0 bias parameter. This non-zero bias parameter determines the state space region in which the terminal dynamics occur. For this simulation, basically this bias parameter is almost zero; hence the terminal dynamics are around the origin. The next section discusses value function solution obtained with the stability based control.

5.3.4 Value Function

To obtain the value function, the position state of the pendulum is observed and a unity binary reward is given if the position state of the pendulum is in the stability region. The control signal is modified bang-bang control signal introduced by (76).

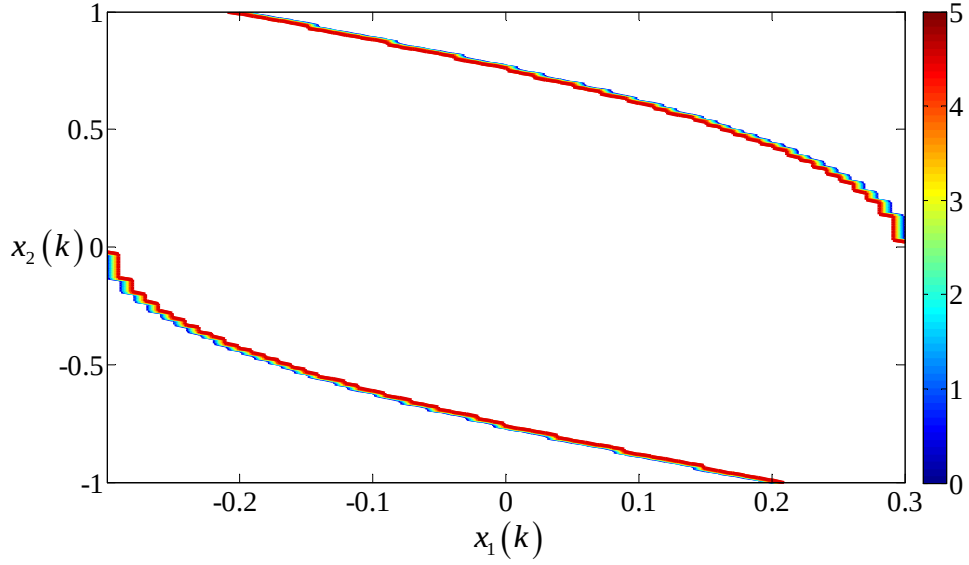


Figure 28: Value function without random control action.

The value function shown by Figure 28 has maximum value of 5 in the stability region as the corresponding discount factor is 0.8. However, outside the stability region, the value function is zero since zero instantaneous reward is provided when $|x_1(k)| > 0.3$. The value function has different behaviour if randomness is introduced in the bang-bang control policy.

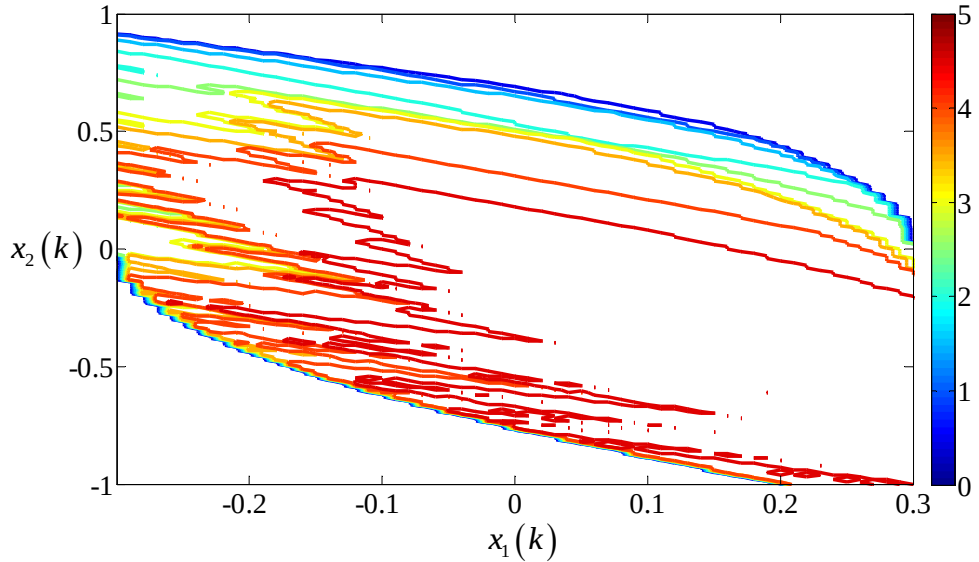


Figure 29: Value function with added random control action.

Since randomness in the control action selection causes the pendulum to move in the opposite direction as shown by the Figure 24, failures occur. Due to existence of

failures, the instantaneous reward is zero and this yields non-maximized value function as shown in Figure 29. This property can be combined with the non-zero trajectory null space analysis addressed in chapters 3 and 4. With respect to the non-zero trajectory null space, it is associated with the higher order parameter and the corresponding basis does not carry any information until the finite horizon sample is reached. It causes bias in the value function unless the finite horizon termination is handled properly. In terms of using this insight, occurrence of non-maximized value function can be associated with the non-zero trajectory null space.

5.3.5 Learned Value Function

The learned version of the value function is analyzed in this section where the instantaneous reward is binary, i.e., unity or zero.

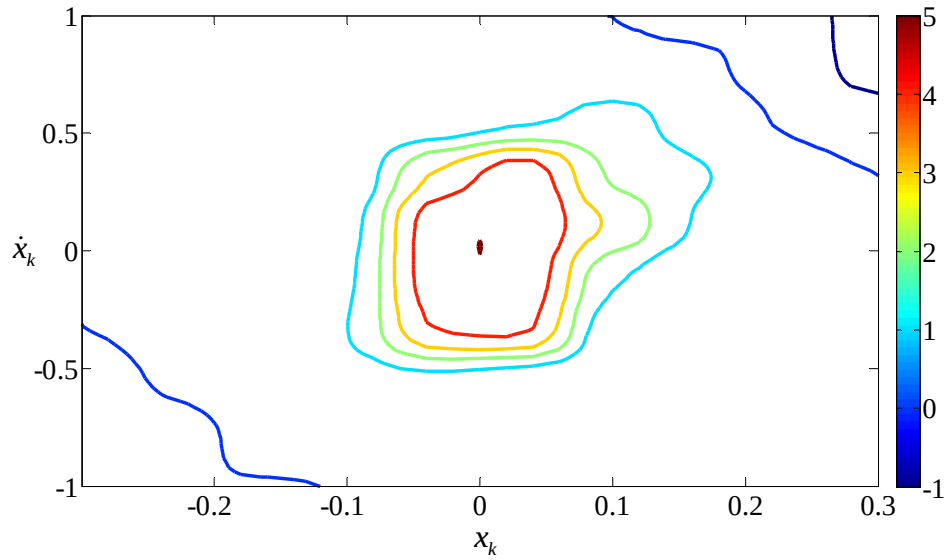


Figure 30: Learned value function with binary reward and bang-bang control.

As can be seen from Figure 30, the maximum value of the contour graph is five and occurs around zero states. In terms of using the gained insights from the value function learning analysis in chapters 3 and 4, the maximum value of the value function can be determined with $V_{\max} = \frac{1}{1-\gamma}$. Therefore, the maximum value function is $V_{\max} = 5$ with the discount factor $\gamma = 0.8$. As the states move away from the zero position/velocity states, then the value function smoothly decreases, even though the same binary reward is provided for $|x_k| \leq 0.3$. This can be explained in terms of the

exploration point of view, digital offset occurring due to chaotic dynamics and finite horizon effect. Since RL algorithm does not explore every state in the state region, it is not expected that the value function will have its maximum value when $|x_k| \leq 0.3$. In terms of the digital offset, it appears in the value function learning because of the chaotic dynamics as addressed in chapter 3. However, since the chaotic dynamics are bounded around $|x_k| = 0.02$ and $|\dot{x}_k| = 0.4$, as can be seen from the Figure 24, the digital offset is limited for these states.

Even though the value function formulation is infinite horizon, in the case of failure it acts like finite horizon. Thus, based on the failure sample, at the edge of the stability region higher order parameters occur and due to existence of the higher order parameters, the value function surface becomes smooth rather than flat in the state space. However, the higher order parameters have no information in their differenced basis until the finite horizon sample is reached. This is known as non-zero trajectory null space as expressed in chapters 3 and 4. Therefore, the higher order parameters cannot be learned until failure due to the non-zero trajectory null space.

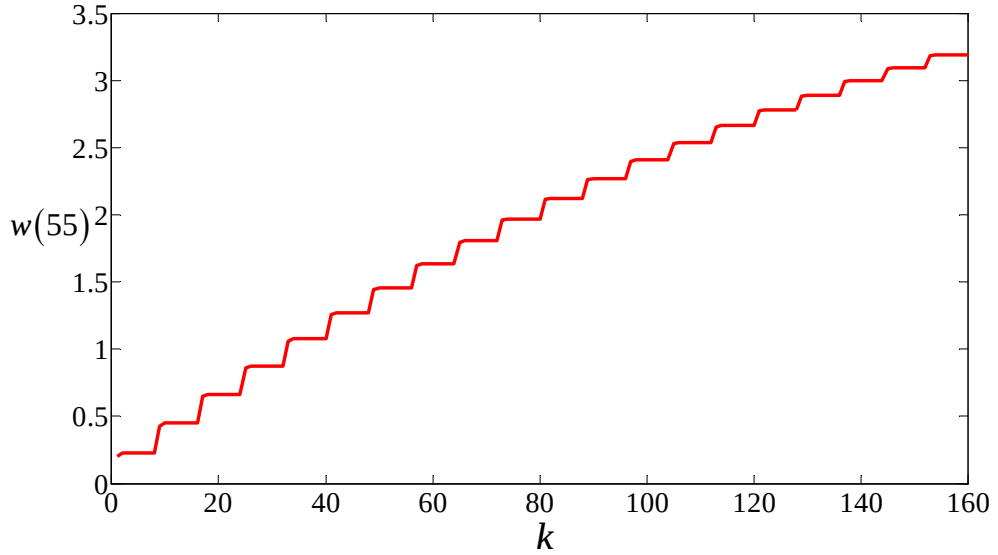


Figure 31: Value function parameter trajectory.

Here $w(55)$ is the 55th parameter trajectory of the value function. Figure 31 also confirms that the higher order parameter of the value function with RBF value function approximation has no parameter update until the finite horizon sample is reached. The higher order parameter has update only at the final sample. This non-

zero trajectory null space slows down the higher order parameter learning process and requires multiple simulation repetition. Insufficient number of simulation repetition causes bias in the value function learning. It is important to note that the higher order parameters analyzed in chapters 3 and 4 occur at the upright position at the terminal sample of the finite horizon or at the saturation threshold whereas for this analysis the higher order parameters occur at the edge of the failure region. In terms of using these insights it can be deduced that the value function might not reach its maximum value in this state region. The next section discusses the chaotic and convergent dynamics and the role of the control signal on these terminal dynamics.

5.3.6 Chaotic and Convergent Dynamics

The learned control policy determines the characteristic behaviour of the states. The Figure 32 shows the states of the pendulum after learning.

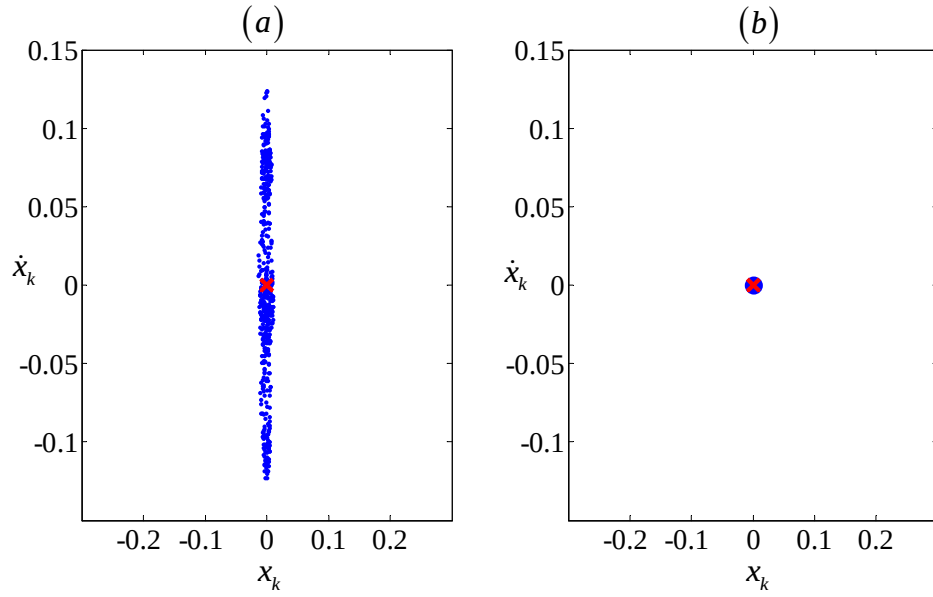


Figure 32: States of the pendulum after learning, a) with bang-bang control, b) with saturated piecewise control.

The red cross is located at the origin and the blue dots are state trajectory. As can be seen from Figure 32a), even after learning, the pendulum oscillates around the origin. Chapters 3 and 4 showed that with bang-bang control, bounded chaotic dynamics occur around the upright unless the initial state yields a zero terminal state. In this simulation, as the initial states are random and also exploration noise exists, the final behaviour is likely to be chaotic. Chaotic dynamics are not periodic, however

bounded, and nearby solutions diverge rapidly and unpredictably in time. They are sensitive to initial conditions; hence small differences/divergences in initial conditions yield a different periodic orbit. As random exploration noise exists in these simulations, it is expected to have different periodic orbits for the chaotic dynamics shown in Figure 32a). However, in chapter 4, it was shown that a saturated piecewise linear control with a sufficiently large saturation threshold ensured that the system converges to the upright. The RL simulation with saturated piecewise control signal in section 5.5 yields convergent dynamics as shown in the Figure 32b). This might increase the learning performance of the RL algorithm which is addressed later in this chapter.

5.4 RL Analysis with Quadratic Reward and Bang-Bang Control

This section initially reviews the LQR based bang-bang control solution and then analyzes the corresponding RL solutions with LQR reward.

5.4.1 LQR Based Bang-Bang Control Solution

The LQR cost function $J(x_k, u_k)$, which is quadratic in state and control signal, is defined as:

$$J(x_k, u_k) = \sum_{t=0}^{\infty} (x_{k+t}^T Q x_{k+t} + u_{k+t}^T R u_{k+t}) \quad (77)$$

where Q is positive definite and R is positive definite or semi-definite to ensure that the cost function is well defined. The bang-bang control law is expressed in terms of the LQR feedback gain, which is reviewed in Appendix C, and the states of the system are:

$$u_k = \begin{cases} -U_{\max} & \text{if } (R + B^T P B)^{-1} B^T P A x_k > 0 \\ 0 & \text{if } (R + B^T P B)^{-1} B^T P A x_k = 0 \\ U_{\max} & \text{if } (R + B^T P B)^{-1} B^T P A x_k < 0 \end{cases} \quad (78)$$

Figure 33 shows the bang-bang control solution and corresponding state trajectory with the control law (78) for the introduced inverted pendulum model.

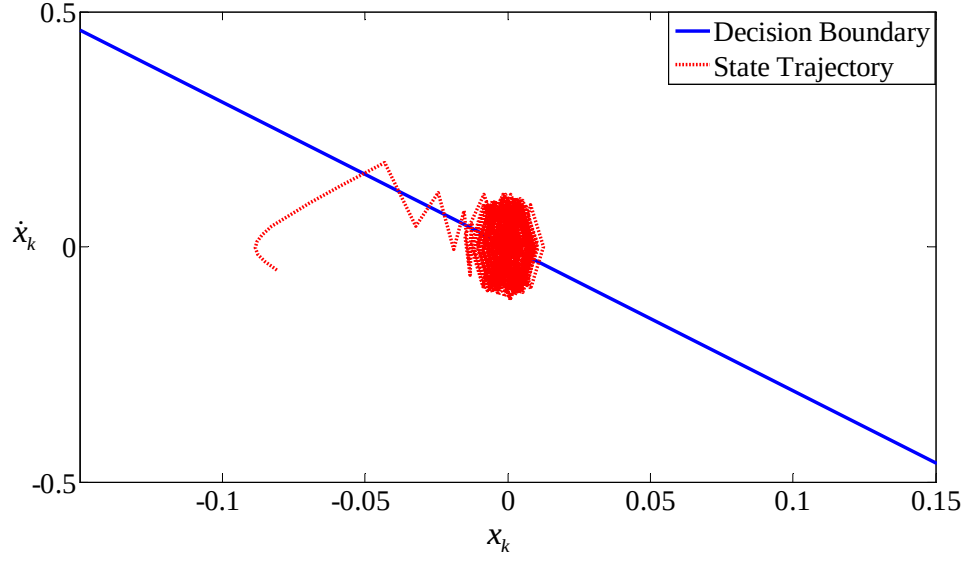


Figure 33: LQR based bang-bang control solution with state trajectory.

As can be seen from Figure 33, the linear bang-bang control solution described by (78) divides the state space into two regions where the control signal is either $-U_{\max}$ or $U_{\max}$. Applying the maximum control signal ensures that the state energy is reduced. Also, for certain initial states, this produces a state trajectory which terminates at the origin. Generally, however, the terminal behaviour is chaotic about the upright as shown for the state trajectory with the initial position state $x_0 = -0.08$ and velocity state $\dot{x}_0 = -0.05$. It is also noticeable that there is digital chattering about the switching line in the state trajectory due to discrete sampling. The next section presents the learned bang-bang control policy which is basically the learned version of the bang-bang solution.

5.4.2 Learned Bang-Bang Control Solution

RL algorithm learns the bang-bang control solution which minimizes the states of the pendulum. The learned solution is basically an approximate solution of the one expressed in section 5.4.1. The reward is LQR, modified as $r(x_k, u_k) = 1 - (x_k^T Q x_k + u_k^T R u_k)$ to have the maximum value of the value function at the origin and the linear discrete pendulum model described in section 5.2.1 is used for this simulation.

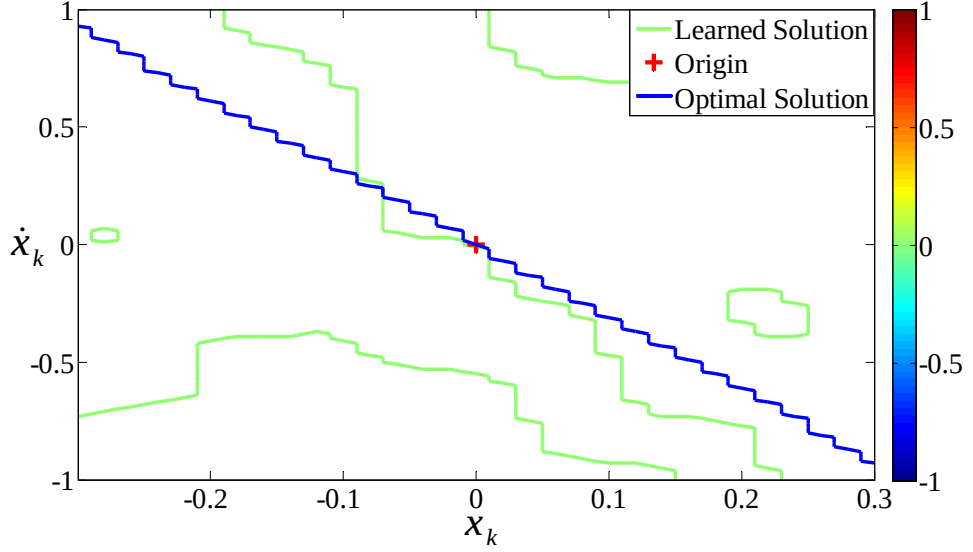


Figure 34: Learned bang-bang control solution with LQR reward.

The red cross is located at the origin. The initial states are $[x_0 \ \dot{x}_0] = [-0.0016 \ -0.0017]$ which are inside the stability region that the RL algorithm with the bounded unity control signal can stabilize. Therefore, the RL algorithm just explores a small state region around the origin with these small initial states and unity control signal. Basically, the RL algorithm generalizes the knowledge gained from this explored region around the origin to the rest of the state space. This RL simulation yields similar learned control policy with the positive position and negative velocity state region whereas the rest of the control region differs. This could be because of successful learning in that particular state region. As can be seen from Figure 34, the learned and optimal decision curves pass through from zero position and velocity states where the terminal dynamics evolve. This reveals that actually the RL algorithm is able to learn the correct solution at the origin where the terminal dynamics occur. There are a couple more curves around the decision curve which occur during the initial wrong learning or failure. It is important to note that the RL with quadratic reward and bang-bang control solution introduces bias in the learned value functions as discussed next.

5.4.3 Value Function

As $u_k^T R u_k$ is a constant and non-zero part of the modified LQR reward $r(x_k, u_k) = 1 - (x_k^T Q x_k + u_k^T R u_k)$, increase in the value function is because of the states

shrinking from their initial values. This constant part of the LQR reward minimizes the learning time since it applies maximum control signal to the pendulum as discussed in section 5.6. However, it causes chaotic dynamics as described in section 5.3.6 and this yields digital offset in the value function. The bang-bang control law (78) is applied to the discrete inverted pendulum and by using the corresponding state trajectory for each initial state the value function is determined.

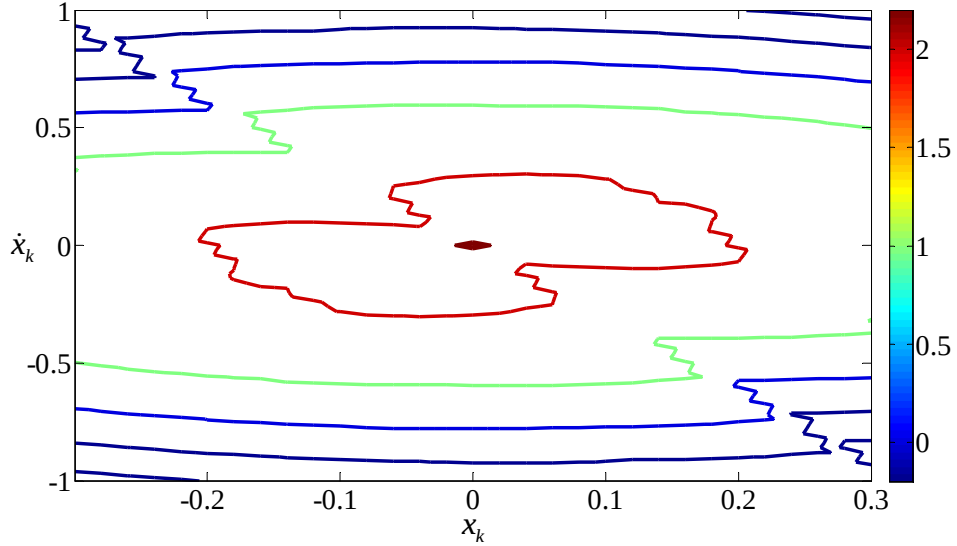


Figure 35: Value function with quadratic reward and bang-bang control.

It is clear from Figure 35 that the bang-bang control solution yields a rough value function surface. This roughness appears because of the bounded chaotic dynamics occurring around the upright position of the pendulum. Since the value function discount factor is 0.8 for 20 samples the value of the discount factor will be insignificant. This number of samples is large enough for chaotic dynamics to affect on the value function even for initial states far from the upright position. However, its impact is much clearer around the origin. The maximum value of the value function is around 2 as can be seen from Figure 35. On the other hand, the maximum value with the discount factor 0.8 and maximum instantaneous reward must be 5. Due to bang-bang control signal, which is the non-minimized part of the instantaneous reward, and also the digital offset occurring because of the chaotic dynamics, the value function cannot be maximized. The reduction in the value function because of the constant bang-bang control signal with $R=0.5$ weighting factor is 2.5 and the reduction because of the chaotic dynamics is around 0.3. Hence, the maximum value of the

value function is around 2. This property can be observed in the learned value function as well.

5.4.4 Learned Value Function

Learning analysis is based on the quadratic instantaneous reward stated in the previous sections and the RL algorithm aims at learning the bang-bang control solution to keep the pendulum upright.

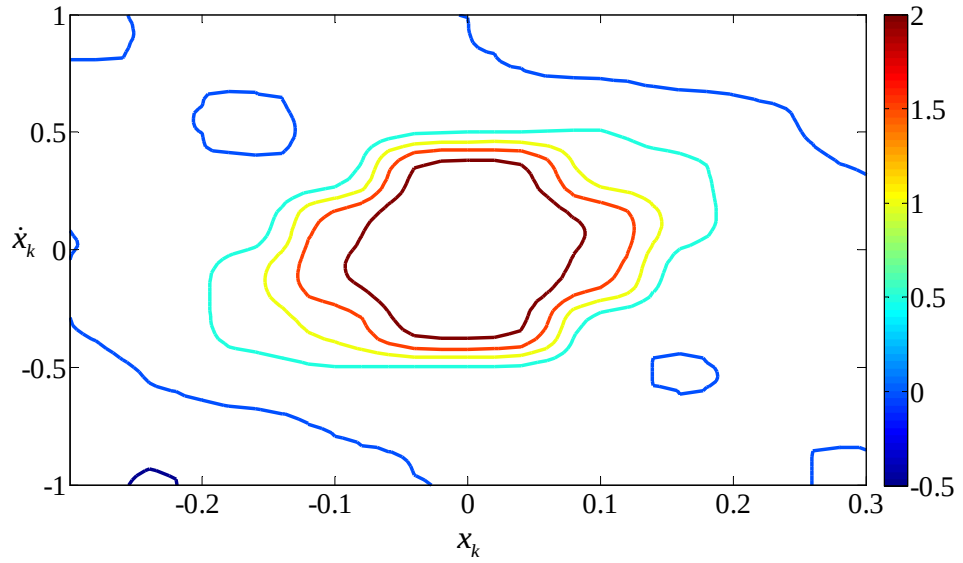


Figure 36: Learned value function with quadratic reward and bang-bang control.

The same argument about the maximum value of the value function is valid for the learned one as well. If the states of the pendulum and the control signal become zero, then the instantaneous reward becomes 1 and this yields $V_{\max} = 5$. However, since the learned control signal is bang-bang and control weighting factor is $R = 0.5$, the maximum value function is $V_{\max} = 2.5$ when the states are zero. Moreover, as expressed in section 5.4.1, bounded chaotic dynamics occur and this yields digital offset around the upright in the learned value function. In addition, the finite horizon effect and non-zero trajectory null space addressed in the section 5.3.5 play a role for the offset in the learned value function. As stated in the section 5.3.5, even though the value function is infinite horizon, failures during the learning process return it to the finite horizon. This generates higher order parameters and because of the non-zero trajectory null space, these higher order parameters cannot be learned until the final sample. Therefore, because of these reasons, the value function does not exactly reach

its maximum value as shown in the Figure 36. The maximum value is around 2 which is consistent with the one shown in the Figure 35.

5.5 RL Analysis with Quadratic Reward and Saturated Piecewise Control

A test problem based value function learning analysis with quadratic reward and piecewise linear control signal is performed in chapter 4 for a first order system. This section analyzes the optimal and learned control policy/value function with the same type of instantaneous reward and saturated piecewise control signal for a higher order system.

5.5.1 Optimal Saturated Piecewise Linear Control Solution

The optimal saturated piecewise linear control signal is defined in terms of the LQR feedback gain derived in Appendix C and the states of the pendulum.

$$u_k = \begin{cases} -U_{\max} & \text{if } (R + B^T P B)^{-1} B^T P A x_k > U_{\max} \\ (R + B^T P B)^{-1} B^T P A x_k & \text{if } \left| (R + B^T P B)^{-1} B^T P A x_k \right| \leq U_{\max} \\ U_{\max} & \text{if } (R + B^T P B)^{-1} B^T P A x_k < -U_{\max} \end{cases} \quad (79)$$

The saturated piecewise linear control signal consists of a saturation part which applies maximum control signal and linear part which reduces the amount of the control signal based on the value of the states.

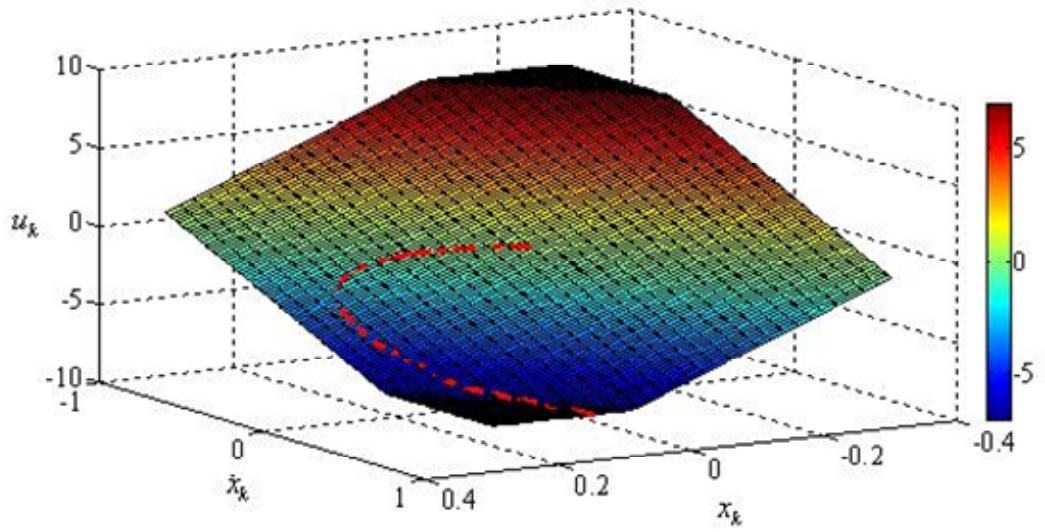


Figure 37: Optimal saturated piecewise linear control signal and state trajectory.

Basically, depending on the value of the feedback gain and the states of the pendulum, the saturated piecewise linear control generates maximum control signal and then as the pendulum approaches the upright position, the control signal linearly reduces as shown in Figure 37. The saturated piecewise control generates smooth control and state trajectories; hence it does not cause chaotic dynamics as shown for the state trajectory of the pendulum with the initial position state $x_0 = -0.15$ and velocity state $\dot{x}_0 = -1$. It is also shown in section 5.3.6 that the terminal dynamics of the pendulum are convergent rather than chaotic. The next section analyzes the saturated piecewise linear control policy learned by the RL algorithm.

5.5.2 Learned Saturated Piecewise Control

In order to eliminate the chaotic dynamics, saturated piecewise linear control is used in the chapter 4 and this section shows that the RL algorithm is able to learn the saturated piecewise control signal when the instantaneous reward is LQR.

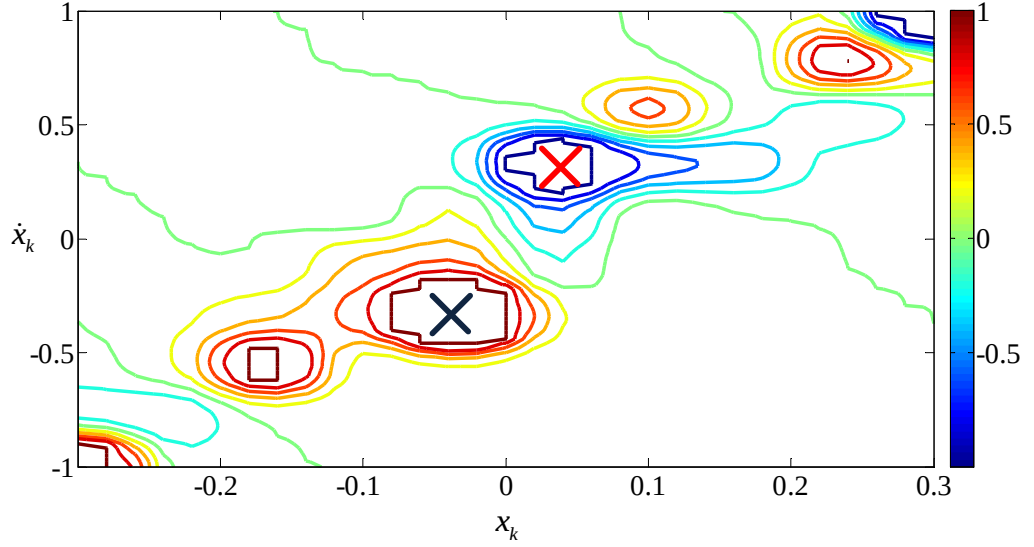


Figure 38: Learned saturated piecewise control.

There are two peaks in Figure 38 around zero position and velocity states. Again, the variations towards the edges of the state region occur during the initial learning and failures. The peak at the blue cross located is the learned maximum positive control signal and the other one with the red cross is the learned minimum negative control signal. It is important to note that the location of the peaks in the learned control policy is consistent with the optimal one given by Figure 37. Between these two peaks there is region where the control signal reduces almost linearly from its maximum value. As this eliminates the chaotic dynamics, optimal and learned value functions do not have rough surfaces as discussed next.

5.5.3 Optimal Value Function

Optimal value function shown in Figure 39 is calculated using the quadratic reward and the optimal saturated piecewise linear control given by (79). The simulation setting and the simulation parameters are the same as described in sections 5.2.1 and 5.2.2.

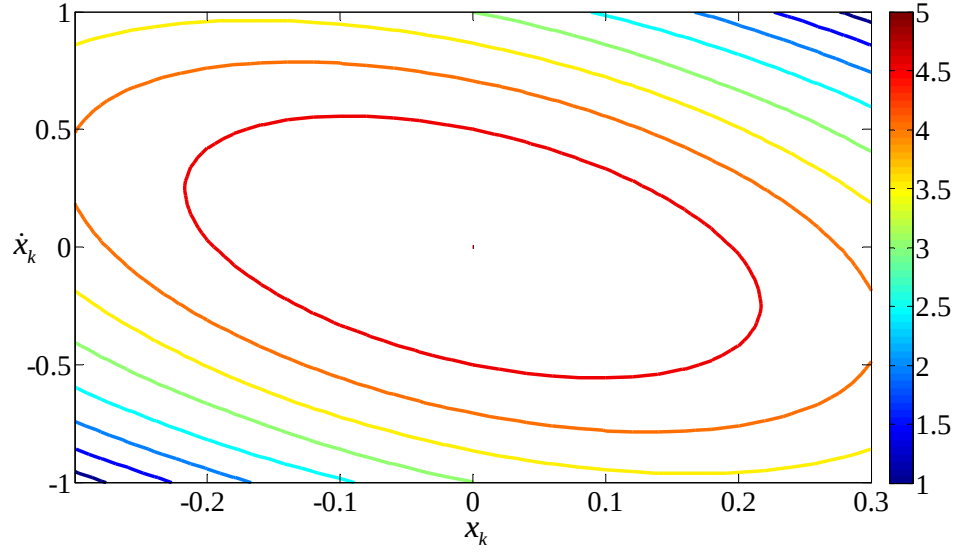


Figure 39: Optimal value function with LQR reward and optimal saturated piecewise linear control.

It is noticeable from Figure 39 that the optimal value function solution with the quadratic reward and optimal piecewise linear control is smooth and the maximum value of the value function occurs around the zero states. This indicates that the optimal piecewise linear control is able to take the pendulum to the upright position without causing chaotic dynamics. Due to linear part of the saturated piecewise linear control signal, the states evolve gradually towards the origin by requiring less control energy. These insights can be confirmed with the learned value function result which is described next.

5.5.4 Learned Value Function

The simulation settings and parameters are the same as the optimal ones addressed previously. The learned value function with the position and velocity initial states -0.0013 is shown in Figure 40.

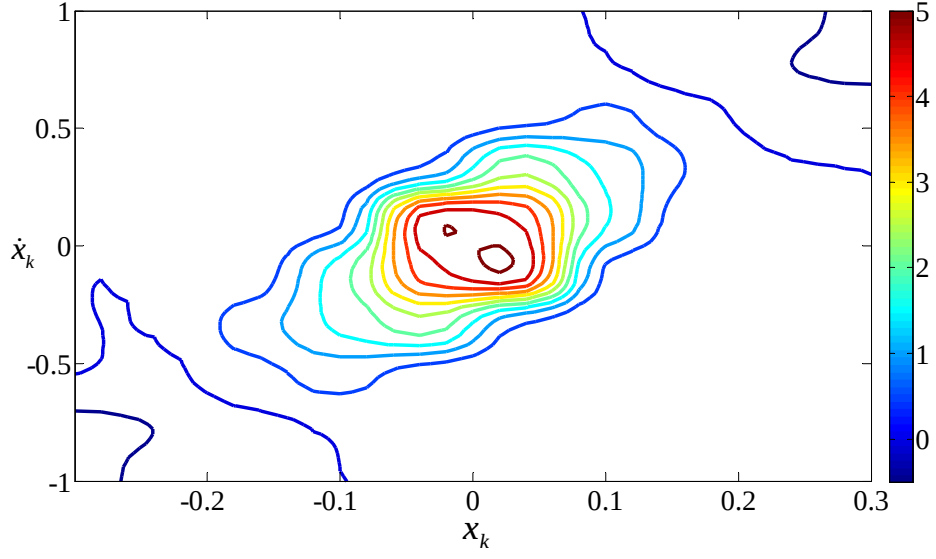


Figure 40: Value function contour graph with LQR reward and saturated piecewise control.

It is noticeable from Figure 40 that the area in which the maximum value occurs is smaller than the previous learned value functions. This can be explained in terms of the role of the saturated piecewise control and quadratic reward. As shown in the section 5.3.6, RL with saturated piecewise control is able to yield final state around zero without causing chaotic dynamics. Thus, the control signal reduces to its minimum as the terminal states move towards the origin where the maximum instantaneous reward is obtained. Furthermore, the value function is maximized around the upright position. This is also same for the optimal solution given by Figure 39.

As derived in chapter 4, the closed form terminal value function solution with the quadratic reward in state/control signal and a saturated piecewise linear control signal has bias, linear, quadratic and higher order parameters. The higher order parameters have non-zero trajectory null space up to the saturation threshold; hence they cannot be learned until the saturation threshold is reached. However, after the saturation threshold their bases gain information and the corresponding higher order parameters are updated. This slows the learning process but the higher order parameters converge to their correct values if enough simulation time is provided. Here the simulation result shows that these higher order parameters together with the bias, linear and

quadratic parameters converge to their correct values; hence the value function reaches its maximum around the zero states. It is noticeable from the optimal and learned value function solutions shown by Figure 39 and Figure 40, respectively, that the direction of the contours is inconsistent. At this stage of this research we are unable to express the reason for this inconsistency.

5.6 Comparison of the Learning Performance from Failures

The number of failures during the learning process might provide insights about the learning performance of the RL algorithm with various instantaneous rewards and control policies.

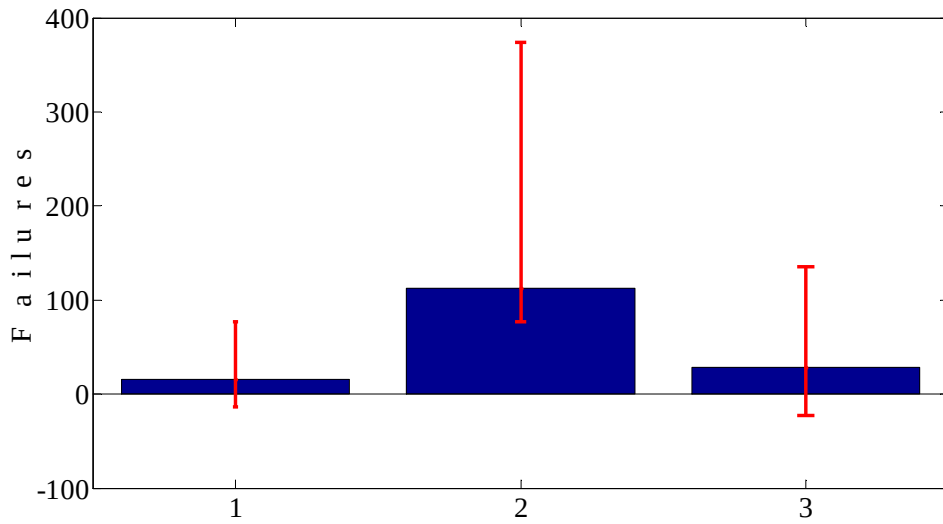


Figure 41: The number of failures for each simulation experiment.

The numbers 1,2,3 on the x axis of Figure 41 represent the number of failures for the RL simulations with binary reward and learned bang-bang control, quadratic reward in state/control signal and learned bang-bang control, quadratic reward in state/control signal and learned saturated piecewise control, respectively. As can be seen from Figure 41, the RL algorithm with quadratic reward and bang-bang control has the largest number of failures whereas the RL algorithm with binary reward and bang-bang control has the lowest number of failures. In order to explain these results, initially consider the exploration/exploitation phase of the learning reviewed in section 5.1.2. Value function learning results in section 5.4.4 show that the learned value function with quadratic reward has larger offset. Since the amount of the

exploration is determined based on the value of the learned value function, RL with quadratic reward has larger exploration noise when the pendulum states bang around the upright. This kind of exploration noise has a negative impact on the instantaneous reward; hence on the learned value function and control policy. Since the performance of the learning is deteriorated by the exploration noise, more failures are expected during the learning process. However, value function reaches its maximum value with the saturated piecewise control; therefore the exploration phase of the RL algorithm disappears and exploitation phase starts. Henceforth, the RL algorithm efficiently uses the knowledge that it gained during the learning process. It is important to note that the RL with quadratic reward and saturated piecewise control has more failures than the RL with binary reward and bang-bang control. This might occur due to the saturation threshold learning process.

5.7 Rate of Convergence Analysis

Since the optimal basis and the correct parameters of the learning problem are not available, it is not possible to follow a generic way for parameter rate of convergence analysis. However, the value function learning and TD error history of the learning problems provide insight about the rate of convergence.

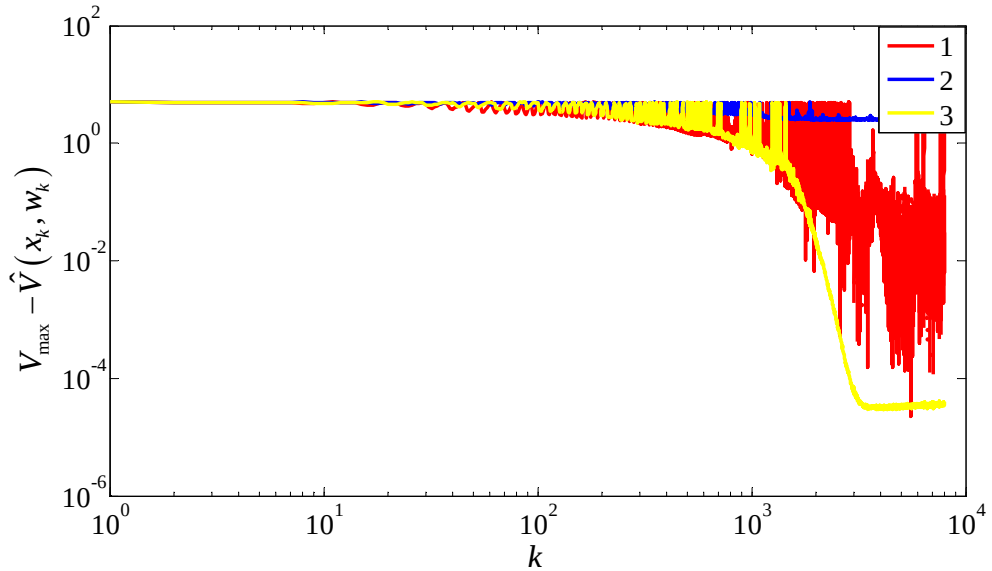


Figure 42: Comparison of the value functions.

The numbers 1,2,3 on the legend Figure 42 represent the value functions of the RL simulation experiments with binary reward and learned bang-bang control, quadratic

reward in state/control signal and learned bang-bang control, quadratic reward in state/control signal and learned saturated piecewise control, respectively. It is noticeable that the RL experiment with LQR reward and bang-bang control has the worst performance in terms of the optimal value function learning. This is basically because of the digital offset, non-zero trajectory null space, and type of the control signal as discussed in the section 5.4.4. On the other hand, RL simulation with LQR reward and saturated piecewise control policy is the fastest one in terms of reducing the gap between the learned and maximum value functions. In the section 5.5, it is noticed that the saturated piecewise control signal does not cause chaotic dynamics; hence the value function does not have digital offset. Therefore, the value function converges to its maximum. The section 5.3 discussed the impact of the digital chattering and finite horizon on the value function when the reward is binary and the control signal is bang-bang. It is highlighted that the value function is not maximized due to chaotic dynamics. Similar insights can be obtained from the TD error history of the RL simulations shown in Figure 43.

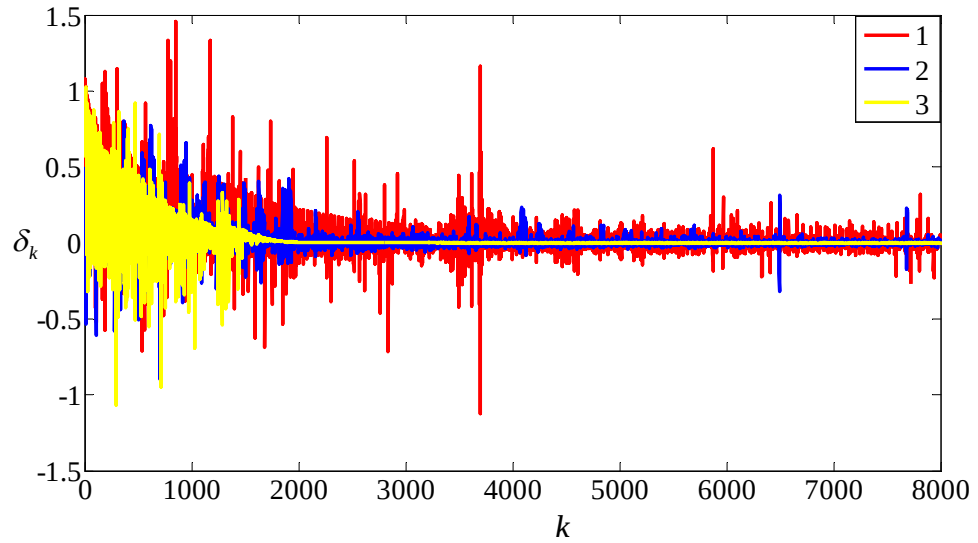


Figure 43: Comparison of the TD errors.

The RL simulation with LQR reward and saturated piecewise control signal reduces its TD error quickly compare to the others. The RL simulation with binary reward and bang-bang control yields the largest TD error due to provided constant reward inside the stability region. However, the LQR reward gets smaller as the states of the pendulum shrink from their initial values.

5.8 Summary

In this chapter, RL algorithm is applied to an unstable inverted pendulum with various types of instantaneous rewards and learned value functions/control policies are compared with their respective optimal solutions. The value function and control policy are approximated by using the RBF approximation and exploration noise is added to learn the control policies. It is shown that the value function cannot be maximized because of the chaotic dynamics which cause digital offset in the value function as extensively examined in chapter 3, too. The impact of the non-zero trajectory null space, which is the key result of the exemplar test problem analysis of chapters 3 and 4, on the value function learning is also highlighted. However, with saturated piecewise linear control, the value function is maximized since the effect of chaotic dynamics on value function learning is eliminated as discussed in chapter 4. It is also shown that the RL algorithm is able to approximately learn the bang-bang and saturated piecewise linear solutions.

6 CPG Based RL Algorithm Learns to Control of a Robot Leg

In this part of the thesis, CPG based RL algorithm is reviewed and this algorithm is applied to a robot leg consisting of two links. As a multi-body modelling software MapleSim is used and this model is combined with the CPG based RL algorithm by using Modelica and Maple software properties. In terms of using Maple multi-body analysis template and Modelica custom component advantages, symbolic inverse kinematics solution for the leg is obtained.

6.1 CPG Controller

Current walking humanoid robots generally utilize pre-calculated walking trajectories determined for a fixed environment. Nevertheless, in practical life it is unrealistic to take into account all possibilities that a HR can face during walking since walking is a dynamic process. Hence, autonomous HRs are designed by equipping them with learning algorithms; so that the robots are capable to cope with complex and uncertain environments [80]. CPG based RL algorithms described next are considered in the literature for autonomous robot walking design.

6.1.1 CPG Control Architecture

CPG basically produces feed forward activation for muscles in spinal cord. The generated pattern can interact with feedback signals, so intrinsic oscillation of the CPG synchronizes its output with feedback input. This is called ‘entrainment’ [34, 36].

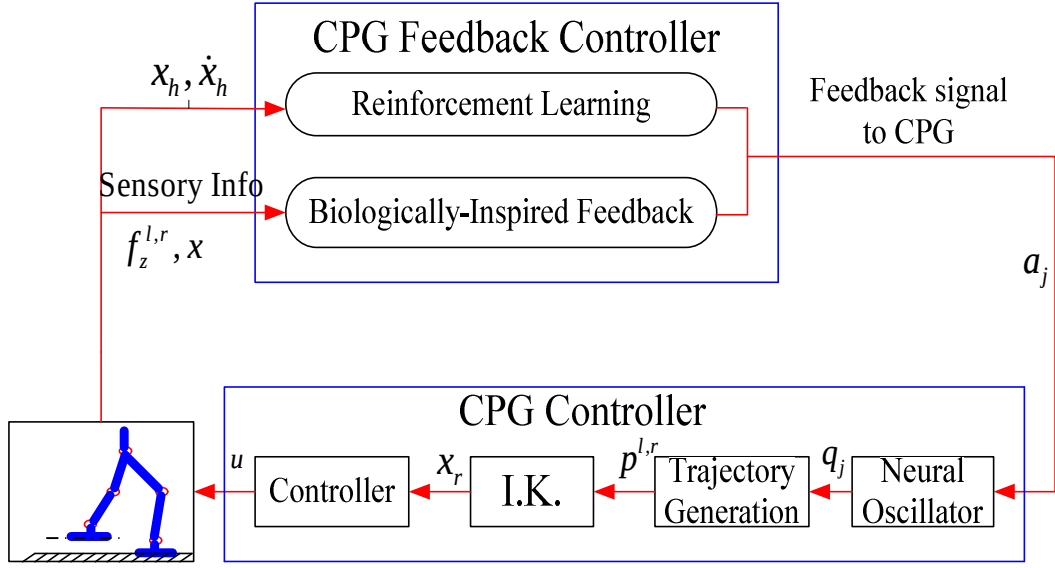


Figure 44: CPG based RL architecture.

In Figure 44, a_j is the feedback signal generated by the RL algorithm and biologically-inspired feedback, q_j is the output of the neural oscillator, $p^{l,r}$ is the right/left leg trajectories, x_r is the reference joint angles for the controller generated by the inverse kinematics (IK) solution, x_h is the angular position of the hip, $\dot{x}_h$ is the angular velocity of the hip and $f_z^{l,r}$ is the vertical ground reaction forces of the right and left foot.

As can be seen from, the CPG control architecture consists of three basic components; CPG controller, robot and CPG feedback controller. In terms of the CPG controller, its first element is the neural oscillator model which adapts the robot to the unstructured environment by using the entrainment property. The output of the neural oscillator is transformed to the corresponding leg trajectories and IK solution converts generated leg trajectories into reference joint angles for the controllers to eliminate the joint position errors. Basically the robot interacts with the environment in terms of using these controller actions. With respect to the CPG feedback controller, RL uses position and angular velocity of the hip to approximate the value function and feedback signal. Moreover, sensory information is utilized for biologically inspired feedback and its output is the input of the neural oscillator model introduced next [36, 81].

6.1.2 Neural Oscillator Model

Matsuoka proposed a neural oscillator model having a number of advantages such as the use of sensory feedback information to adapt its internal dynamics to the changing environment [82]. The mathematical form of the neural oscillator model is defined as:

$$\tau_{CPG} \dot{z}_j = -z_j - \sum_{k=1}^n w_{jk} q_k - \gamma_n z_j' + c_n + a_j \quad (80)$$

$$\tau_{CPG} \dot{z}_j' = -z_j' + q_j \quad (81)$$

$$q_j = \max(0, z_j) \quad (82)$$

where z_j is the firing rate (mean membrane potential), z_j' is the variable that is used for self inhibition of the neuron modulated by the adaptation constant γ_n , w_{jk} controls the mutual inhibition (inhibitory synaptic weight from k^{th} neuron to j^{th} neuron). The tonic input c_n plays a key role to determine the amplitude of the oscillation, a_j is the feedback signal from the CPG feedback controller, τ_{CPG} is the time constant for the mean membrane potential, and τ_{CPG}' is time constant for the adaptation effect of the neuron. These time constants determine the speed and shape of the oscillator output. The Figure 45 shows components of the a neural oscillators with two inputs and two outputs.

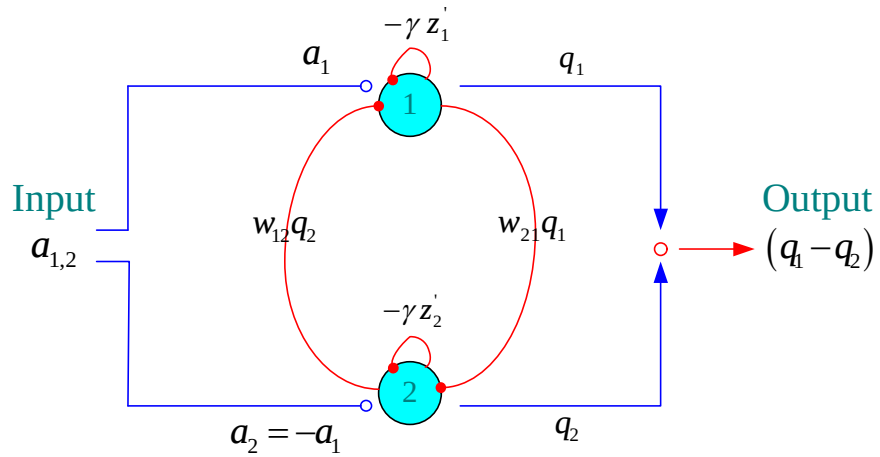


Figure 45: Configuration of the neural oscillator model.

Here, $a_{1,2}$ is the input of the neurons, and $q_{1,2}$ is the output of the neurons. If a connection are terminating with a filled circle, then it has an inhibitory effect on the other neuron, otherwise it has excitatory effect on the other neuron.

In order to reveal the entrainment property of the neural oscillator, a simulation is performed. The length of the simulation is 18 seconds, where both frequency and amplitude of the input sine signal are 0.4 and applied between 4.1 seconds to 14 seconds, the time constants are $\tau_{CPG} = 0.224$, $\tau'_{CPG} = 0.280$, the weights between the neurons are $w_{12} = w_{21} = 2$, the adaptation constant is $\gamma_n = 2.5$, initial states for the mean membrane potential are $z_1 = 0.8308$, $z_2 = -0.0941$, initial states for the self-inhibition effect are $z'_1 = 0.7136$, $z'_2 = 0.1770$ and the tonic input is $c_n = 2.36$.

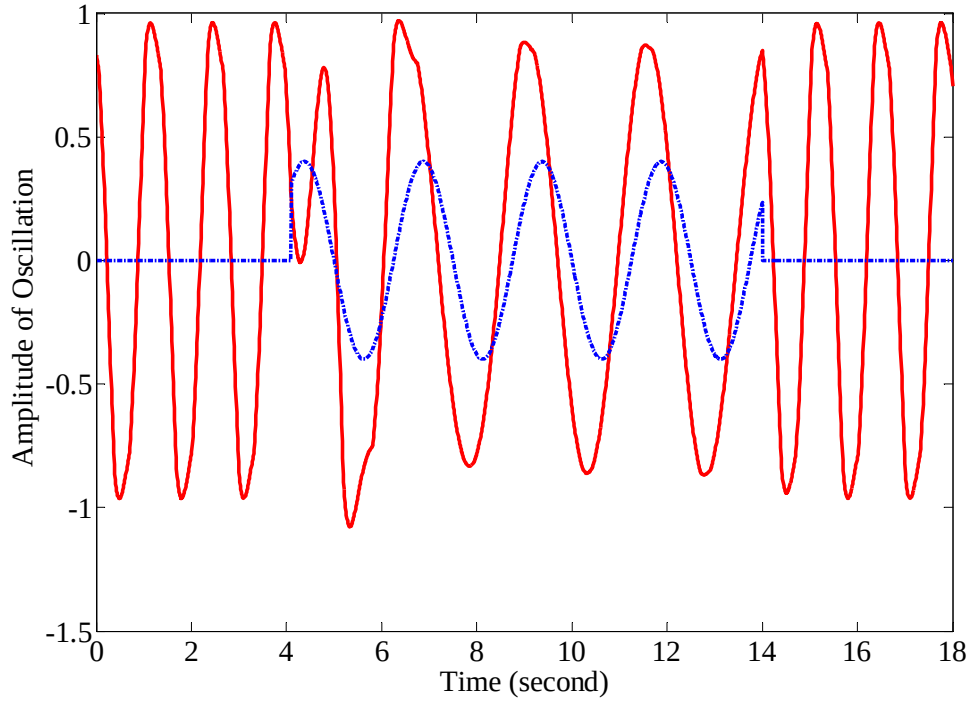


Figure 46: The neural oscillator model entrainment property [36].

As can be seen from Figure 46, the output frequency and the amplitude of the neural network is changing based on the amplitude and frequency of the sinusoidal input signal. It is clearly shown that the phase difference between the input and output of the neural oscillator model are locked due to entrainment property and this property always works as long as the amplitude of the input is sufficiently large and the input

frequency is close to the natural frequency of the oscillator. The frequency of the oscillator entrained with the input frequency and the phase difference between the input and output is consistent between 4.1-14 seconds. Hence, it can be deduced that internal dynamics of the oscillator are be modulated by the input signal [36, 59, 81].

6.1.3 Leg Trajectory Generation

Position of the legs in the z (vertical) direction is specified in terms of the neural oscillator output.

$$p_z^l = Z_0 - A_z (q_1 - q_2) \quad (83)$$

$$p_z^r = Z_0 + A_z (q_1 - q_2) \quad (84)$$

where Z_0 is offset position for the leg movement in vertical direction, A_z is defined to scale the amplitude of the neural oscillator output, p_z^l is the vertical position of the left leg and p_z^r is the vertical position of the right leg. The position of the legs in the horizontal x (forward) direction is specified as:

$$p_x^l = X_0 - A_x (q_1 - q_2) \quad (85)$$

$$p_x^r = X_0 + A_x (q_1 - q_2) \quad (86)$$

where X_0 is offset in the forward direction, A_x is scaling factor or gain for the oscillator output and p_x^l is the horizontal position of the left leg and p_x^r is the horizontal position of the right leg. These offsets and scaling factors are determined based on the length of the legs and maximum value of the neural oscillator outputs [36, 59, 81]. After obtaining the leg trajectories, the next target is to convert them to the reference joint angles by using IK solutions.

6.1.4 Inverse Kinematics Solution

IK is a tool used in robotics to determine relative joint angles of the legs based on position of each leg in Cartesian coordinates. A number of methods have been proposed such as pseudo-inverse, Jacobian transpose, Levenberg-Marquard, damped least square and quasi-Newton to solve the IK problem. It basically aims at tracking the target position of each leg.

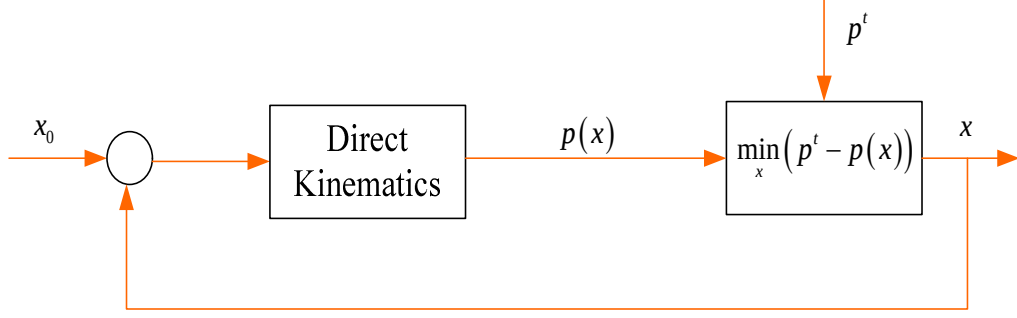


Figure 47: Optimization method based on IK solution.

Assume that a multi-body system (MBS) has n joints and each joint angle is specified as x value. If the MBS has k end-effectors, then the position of each end-effector is denoted as $p(x_1), p(x_2), \dots, p(x_k)$ which is a function of the joint angles and target position of the k end-effectors are represented as $p_1^t, p_2^t, \dots, p_k^t$. The target of the IK solution is to determine the joint angles that satisfy the target position by using a loop as defined in Figure 47.

$$p_i^t = p(x_i) \quad (87)$$

However, IK might not have any feasible solution or a unique solution which satisfies the target position. In this case iterative methods can be considered to obtain an approximate solution for the desired solution. The position p_i is linearly approximated by using the Jacobean matrix.

$$J(x) = \left(\frac{\partial p_i}{\partial x_j} \right)_{i,j} \quad (88)$$

Jacobian matrix is also function of the x joint angles. As the current values of x, p, p^t are known, $J(x) = J$ is calculated and then the joint angles are updated.

$$x_{k+1} = x_k + \Delta x \quad (89)$$

In order to determine the incremental value Δx , Jacobean transpose approach is considered to avoid from singularity problem occurring with inverse Jacobean. The value of increment Δx is defined as:

$$\Delta x = \alpha J^T e \quad (90)$$

where e is the position error $e = p^t - p(x)$ and α is a scalar learning value selected as:

$$\alpha = \frac{\langle e, JJ^T e \rangle}{\langle JJ^T e, JJ^T e \rangle} \quad (91)$$

The learning rate given by (91) is capable to adapt its value based on the error and Jacobean matrix [83, 84]. The Table 1 summarize the inverse kinematics solution.

Assign initial joint angles x_0 , target error e and target position p^t .
Determine the current position $p(x)$ and Jacobean $J(x)$ of each end-effector by using the sensory information.
Find the position error by using the target and current position.
$e = p^t - p$
Compute the α learning rate in terms of the Jacobean and position error.
$\alpha = \frac{\langle e, JJ^T e \rangle}{\langle JJ^T e, JJ^T e \rangle}$
Update the joint angles by using Jacobean, position error and learning rate.
$x_{k+1} = x_k + \alpha J^T e$
Check the error criteria. If it does not satisfy the target error, then go to step 2.
Stop the iteration.

Table 1: Pseudo-code of IK solution with Jacobean transposes method.

6.2 CPG Based RL Modelling in MapleSim and Modelica

A key feature of the MapleSim software is to allow users to create their own Modelica or Maple models to design various learning/control algorithms and these generated models can be combined with MBS models. MapleSim is able to generate C-code of the whole model and also it is still possible to use animation property of the software. Thus, motion of the designed robots can be depicted visually. The CPG based RL algorithm applied to the robot leg in MapleSim is shown in Figure 48.

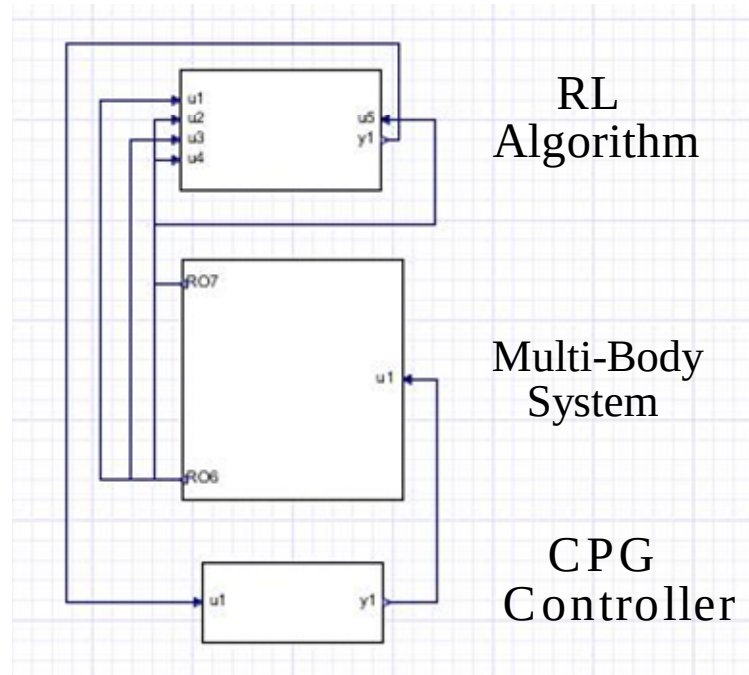


Figure 48: CPG based RL algorithm model in MapleSim.

There are three main components in Figure 48; RL algorithm, CPG controller and multi-body system. Each block represents one part of the CPG based RL algorithm. Modelica code for re-initialization of the leg is shown in Table 2.

```

model reInitRobotLeg
  extends Maplesoft.Icons.CustomComponent;
  Real omega;
  Real omega1;
  Boolean on;
  parameter Real failureStatephi1Global;
  ...
  Modelica.Blocks.Interfaces.RealInput phi annotation
    (Placement(transformation( extent = { {-110, 52}, {-90, 72} }, rotation = 0)));
  ...
equation
  on=abs(phi)>failureStatephi1Global or abs(phi1)>failureStatephi2Global;
  omega=der(phi); omega1=der(phi1);
  when on then
    reinit(phi,phi1Global);

```

```

...
    reinit(omega1, omega2Global);
end when;
annotation (uses(Modelica(version = "3.1")));
end reInitRobotLeg;

```

Table 2: Modelica code to re-initialize the robot leg.

As can be seen from the Table 2, the code is starting with the specification of the model name and then the model parameters are introduced by specifying the type of the parameters such as real, parameter real. The next step is definition of the inputs and outputs of the reInitRobotLeg model. Even though the robot leg with two joints has four states where two of them are position states and two of them are the velocity states, introducing just two inputs for this model is enough since der() Modelica command gives velocity of the corresponding position state. There is no need to define output for this model as the defined inputs act like output as well. In the equation part of the model, various type of equations such as differential, algebraic and equations including ‘for and while’ loops, ‘if and when’ conditions, or other Modelica functions/commands can be used. For example, Modelica code for feedback signal parameter update has a ‘for’ loop, ‘when’ condition together with sample() and pre() commands.

```

model feedbackSignalParamUpdate
  extends Maplesoft.Icons.CustomComponent;
  parameter Real na;// "Actor learning rate";
  parameter Integer sn;//"The number of basis in each axist";
  Real np[1,1]; Boolean clock;
  Modelica.Blocks.Interfaces.RealInput preWa[sn * sn,1] annotation
    (Placement(transformation(extent = {{-110,102},{-90,82}}, rotation = 0)));
  ...
  Modelica.Blocks.Interfaces.RealOutput preWaUpdate[sn*sn,1] annotation
    (Placement(transformation(extent = {{90,82},{110,62}}, rotation=0)));
  ...
initial equation
Wa=fill(0,sn*sn,1); preWa=fill(0,sn*sn,1);

```

```

equation
np[1,1]=(n[1]);
Wa = preWa + bPen*(na) * (TDError) * np;
Wavector[sn*sn] = Wa[sn*sn,1];
clock = sample(0, 0.01);
when clock then
    for i in 1:sn*sn loop
        preWaUpdate[i,1] = pre(Wa[i,1]);
    end for;
end when;
end feedbackSignalParamUpdate;

```

Table 3: Modelica code for RL feedback signal parameter update.

In Table 3, together with the model inputs, model outputs are defined as well. A further difference in this model is the initial equation part considered to assign initial values for the current and previous values of the parameter vector. This step is necessary to start the simulation with the same parameter vector for each simulation run. Otherwise, MapleSim initializes the vector randomly or just guesses initial parameters and this yields inconsistent simulation results. To use the previous parameter vector at the current time, `sample()` and `pre()` functions are used. The `sample()` function holds the previous value for 0.01s for this simulation and `pre()` function assigns this value as the previous value [80].

6.3 CPG Based RL Application to Single Robot Leg

In this section, a simulation is performed to analyze the learning results of the CPG based RL algorithm. A robot leg which is basically a double inverted pendulum is modelled by using the MapleSim multi-body modelling software. The CPG based RL algorithm is applied to this model using the Modelica and Maple software. The learning results for the CPG based RL algorithm are presented.

6.3.1 Simulation Parameters

The CPG based RL algorithm is applied to single leg of a robot with two joints where the target is to keep the leg upright by generating appropriate control signals for each joint.

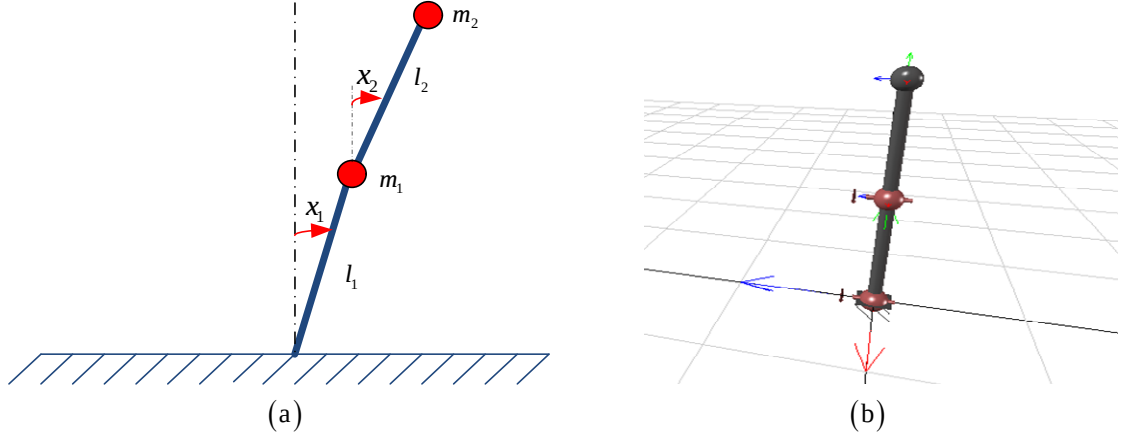


Figure 49: Leg model, a) robot leg frame, b) MapleSim 3D visualization.

In Figure 49, x_1 is the position state of the first joint, x_2 is the position state of the second joint, l_1 is the length of the first joint, l_2 is the length of the second joint, m_1 is the mass for the first joint and m_2 is the mass for the second joint, where all the masses and lengths are unity for this simulation experiment. The dynamical equation of the double pendulum model is given in Appendix D and coupled neural oscillator parameters are the same as described in section 6.1.2. The initial position states of the leg for the first and second joints are -0.004 and -0.0012 , respectively, and initial velocity states for the first and second joints are 0.002 and -0.001 , respectively. The vertical offset and scaling values of the leg are $A_z = Z_0 = 0.2955$, the horizontal offset value of the leg is $X_0 = 1.9553$ and the corresponding scaling factor is $A_x = 0.0447$.

RBF based approximation introduced in the section 5.2.2 is used to approximate the value function and control feedback signal. The input states of the RBF are coming from the first joint of the leg where the lower and upper boundaries of the position state are -0.3 and 0.3 , respectively, the lower and upper boundaries of the velocity state are -2 and 2 , respectively. The discount factor of the RL algorithm is 0.8 , value function and feedback signal learning rates are 0.6 and 0.9 , respectively.

Standard deviation of the noise is $\sigma = 0.5$ where the noise is random Gaussian. The RL outputs which are the inputs for CPG are defined as:

$$a_1 = -a_2 = a_{\max} \left(\tanh \left(\phi(x)^T w_k^a \right) \right) \quad (92)$$

where $a_{\max} = 4$. Proportional PID gains for the first and second joints are $kp_1 = 32$ and $kp_2 = 50$, respectively, integral time constants for the first and second joints are $ti_1 = 145$ and $ti_2 = 96$, respectively, derivative time constants for the first and second joints are $td_1 = 4$ and $td_2 = 6$, respectively. The simulation is re-initialized when the position states of the leg are $|x_1| > 0.3$ and $|x_2| > 0.6$. The instantaneous reward is designed in terms of the joint position as:

$$r(x_k) = 0.5 \frac{\cos(x_1) - \cos(x_1^f)}{1 - \cos(x_1^f)} + 0.5 \frac{\cos(x_2) - \cos(x_2^f)}{1 - \cos(x_2^f)} \quad (93)$$

where $x_1^f = 0.3$ and $x_2^f = 0.6$ are failure states for the first and second joints, respectively. Basically the reward is maximized when both joint positions converge to the upright. Learning results presented below show that the CPG based RL algorithm is able to obtain maximum instantaneous reward.

6.3.2 Neural Oscillator Output and Learned Leg Trajectories

As the trajectory of the leg is determined based on the neural oscillator output, it is important to show that the neural oscillator output is converging to a value, so that the leg is upright. In order to avoid having larger leg trajectories than the actual leg length, the output of the neural oscillator given by (82) is adapted as $q_j = \min(1, \max(0, z_j))$; hence the maximum value of the neural oscillator output is unity when the appropriate learning occurs as shown in Figure 50.

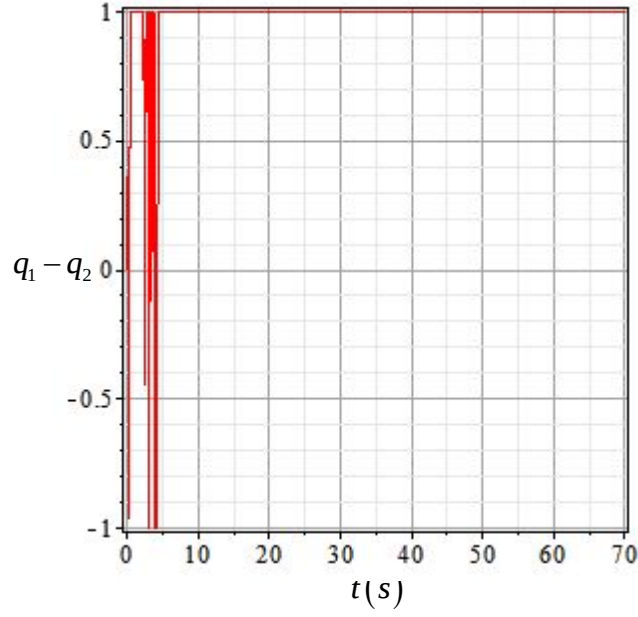


Figure 50: Learned neural oscillator output.

The neural oscillator output initially oscillates between -1 and 1 and then it converges to its maximum value as shown in the Figure 50. Basically, initial oscillation occurs during the learning process. This learning process depends on the RL part of the algorithm since the input of the neural oscillator is output of the RL algorithm. As a result of this convergence in the neural oscillator output, the leg trajectories converge to the values where the leg is upright as shown in Figure 51.

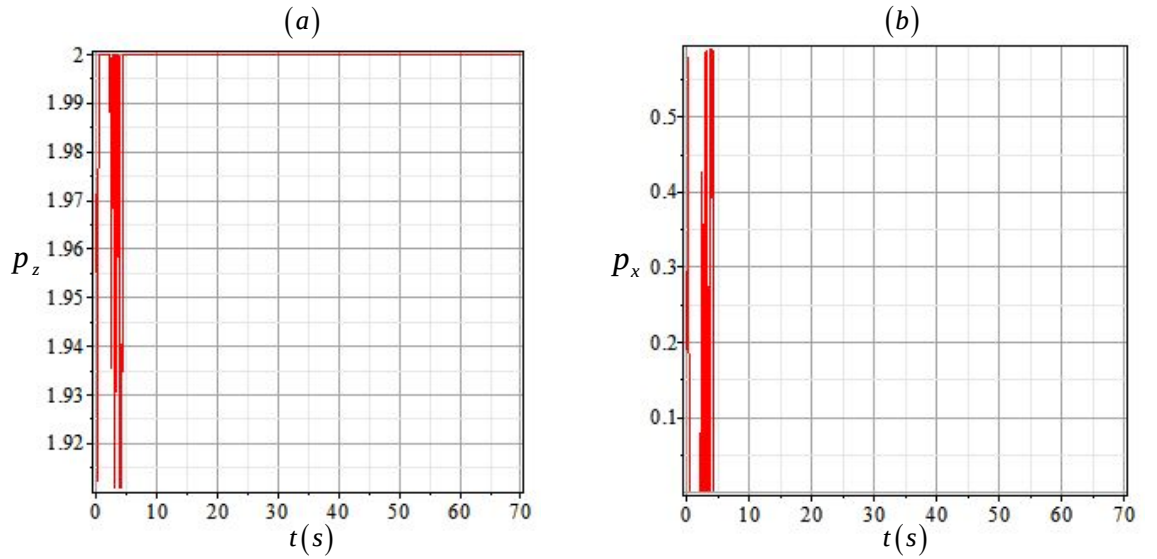


Figure 51: Learned leg trajectory, a) vertical position, b) horizontal position.

The instantaneous reward expressed in the section 6.3.1 reaches its maximum value when the leg is upright. As the key goal of the RL algorithm is to maximize this reward, the RL algorithm learns to generate appropriate neural oscillator input; so that the leg is upright. The length of the leg in the z direction converges to 2, which is the maximum length of the leg that it can converge in the z direction, and the length of the leg in the x direction is around zero as can be seen from the Figure 51. The next step is to generate reference joint position states for the PID controllers in terms of using the learned tip point trajectory of the leg.

6.3.3 IK Solution and Reference Position States

IK solutions are usually obtained in terms of numerical methods which are computationally expensive and also information loss about the motion of the dynamic system occurs. The MapleSim multi-body modelling software offers symbolic IK solution to eliminate the computational burden and information loss issues of the numerical methods. In order to ensure that the target elements appear in the IK solution, constraints need to be introduced by choosing appropriate options for each joint. Thus, initially a spherical joint and a fixed frame is connected to the end of the robot leg, as shown below, to reduce the 2 degree of freedom (DOF) leg to zero DOF. Then, all the rotational initial conditions for the joints and rigid bodies are ignored, so the constraints are included in the model [85].

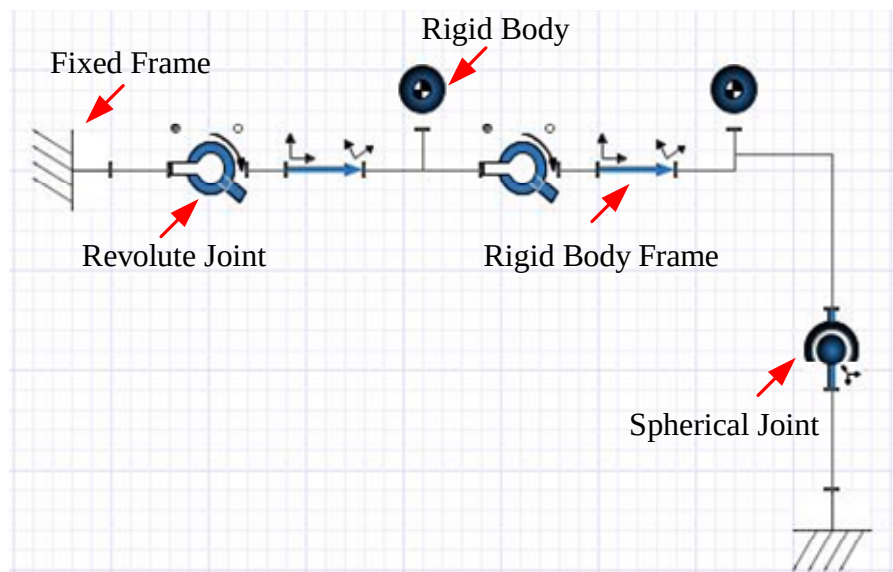


Figure 52: The leg model with reduced DOF.

The fixed frame references connected to the revolute joint are all zero while the fixed frame references connected to the spherical joint are the symbolic target positions in Figure 52. The next step is to use multi-body analysis template of the MapleSim to extract the equations of the leg.

```
mModel:=MapleSim:-LinkModel();
```

```
MB:=mModel:-GetMultiBody(simplify=true);
```

To get the constraints:

```
vPosCons:=MB:-GetPosCons();
```

Finally, solutions for each position state are obtained as:

```
theta1_sol:=solve({vPosCons[1].vPosCons[2]},{R1_theta(t),R2_theta(t)},Explicit);
```

The next step is to generate IK model in terms of using the Maple custom component or the Modelica custom component options. In Appendix D, the joint equations of the leg and IK model generated by using Modelica custom component are given. The inputs of the model are the target position for the leg which is basically the tip position of the leg in vertical and horizontal directions and the outputs are the reference joint angles for each joint of the leg.

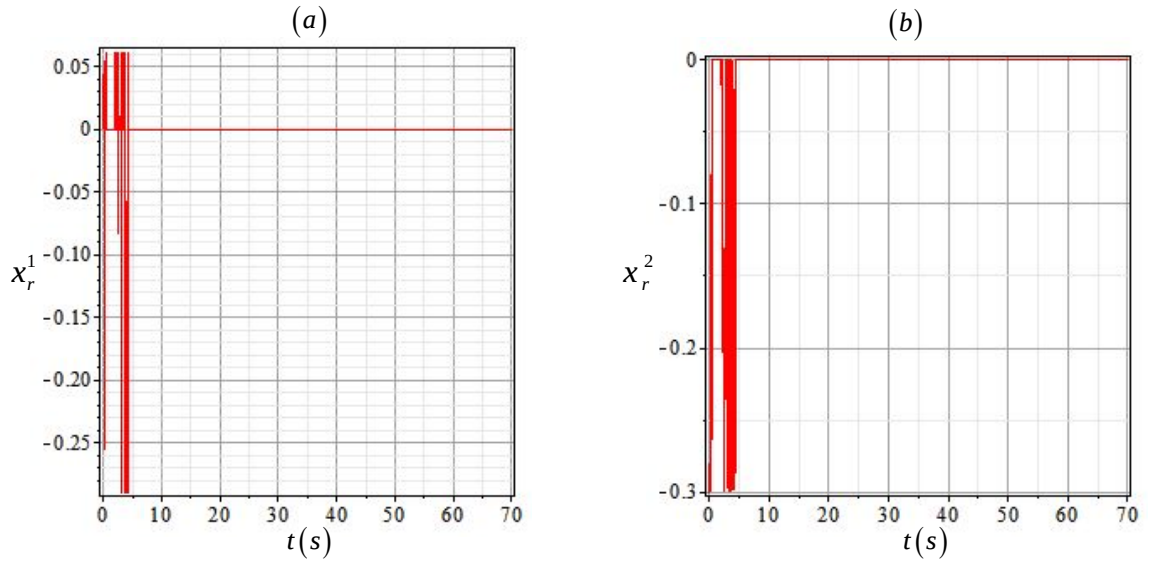


Figure 53: Learned reference trajectories, a) for the first joint, b) for the second joint.

Here x_r^1 and x_r^2 are the reference trajectories for the first and second joints. During the learning process initially, generated reference trajectories evolve in a state region and then converges to zero as shown in the Figure 53 [86]. The convergence in

reference position trajectories occur as a result of learning appropriate feedback signal a_j which depends on the learned optimal value function $\hat{V}(x_k, w_k)$ addressed next.

6.3.4 Learned Value Function and TD Error

A key aspect of assessing the performance of the RL algorithm is the value function learning analysis. Obtaining the optimal value function in terms of attaining the maximum instantaneous reward leads to zero TD error.

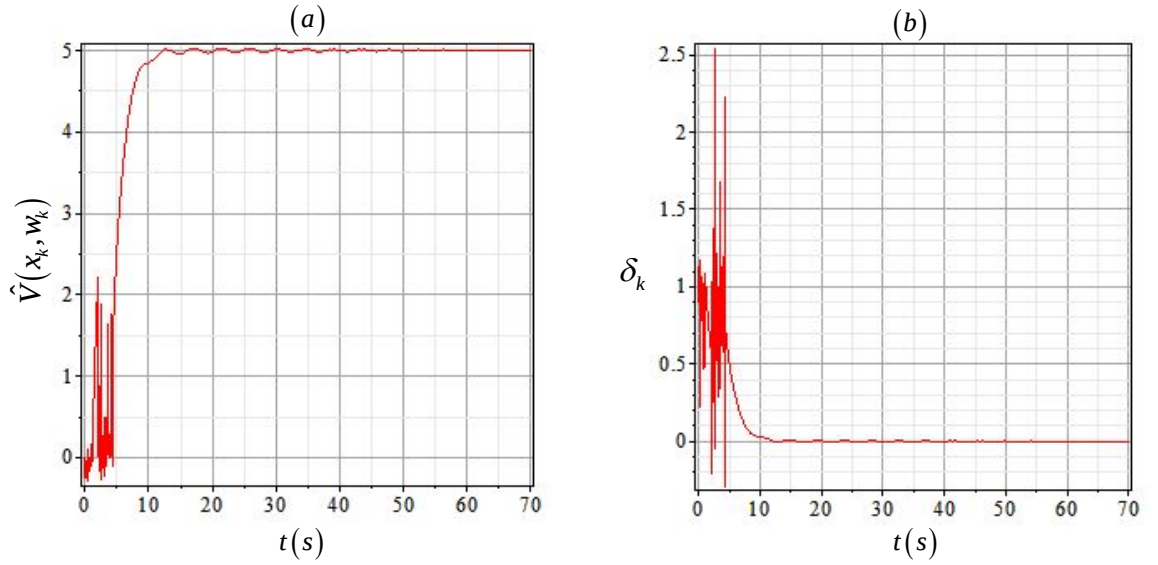


Figure 54: a) Learned value function, b) TD error.

As 3D plots are not available with the MapleSim multi-body modelling software, learned 2D value function with respect to time is shown in Figure 54a). Since the learning discount factor is $\gamma = 0.8$, the maximum value of the value function is $\hat{V}_{\max}(x_k, w_k) = 5$ when the maximum instantaneous reward is obtained. Basically, maximum value of the instantaneous reward and value function occur around the upright position of the leg. During the initial learning process, the RL algorithm is unable to keep the leg upright due to insufficient learned knowledge; hence the value function has a small value. However, as the optimal learning is achieved, the value function reaches its maximum value. It might be important to note here, initial fluctuation in the learned value function disappears since the exploration phase of the learning ends and exploitation phase starts. Together with the optimal value function learning, the TD error becomes zero as shown in the Figure 54b). This means that the parameter update process for the value function and RL control policy part will

terminate. As a result of these convergences, the terminal dynamics of the leg will converge to zero as described next.

6.3.5 Terminal Dynamics of the Leg

The main goal of the CPG based RL algorithm is to stabilize the leg at the unstable equilibrium where the terminal dynamics of the leg are all zero.

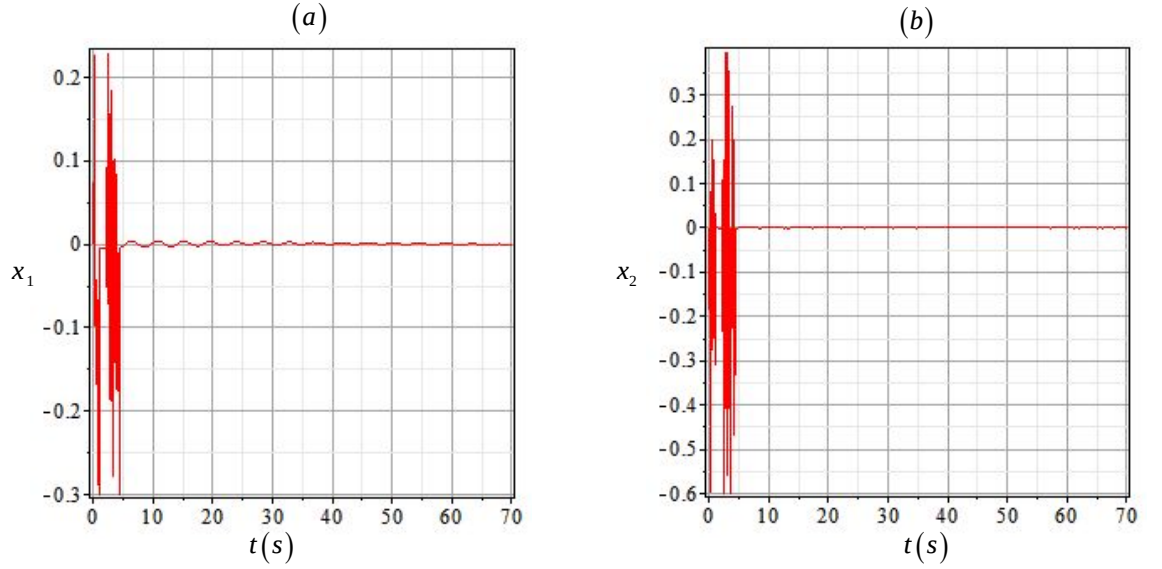


Figure 55: Position trajectory of the leg, a) for the first joint, b) for the second joint.

The position states of the leg for the first and second joints are used to form the instantaneous reward. Therefore, both the instantaneous reward and the value function are maximized when the position states of the leg have the desired values. As can be seen from the Figure 55, the both joint position states are converging to the zero states after initial failures.

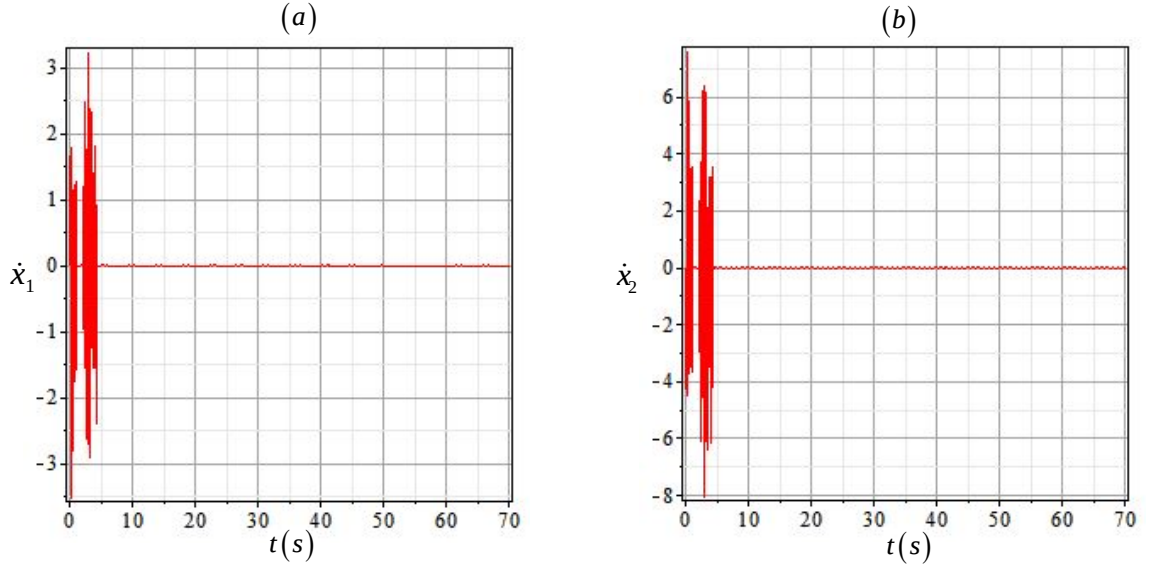


Figure 56: Velocity trajectory of the leg, a) for the first joint, b) for the second joint.

Convergence to the upright can also be confirmed from the velocity states of the leg. Together with the position states of the leg, velocity states converge to zero, as shown in the Figure 56, and this leads to minimum control effort.

6.4 Summary

In this chapter, initially, CPG based RL algorithm is reviewed. Then, the leg of the robot is modelled by the MapleSim multi-body modelling software and the CPG based RL algorithm is applied to this model in terms of using the Modelica and Maple software properties. The symbolic IK solution is obtained by using Maple and Modelica advantages, so that the disadvantages of numerical IK solutions are eliminated. It is shown that the CPG based RL algorithm is able to learn the appropriate control actions for the both joints of the leg which take the leg to the upright position and keep it at that position. The learned neural oscillator output, value function, leg trajectories, target position states and terminal dynamics of the leg are presented and analyzed.

7 Conclusion and Future Work

This PhD thesis presents a generic approach for the parameter convergence and rate of convergence analysis of the TD(0) and residual gradient learning algorithms. In the literature, it is reported that the learning process of the TD algorithms are slow; however, the reason behind this sluggish learning has not been clearly addressed. Moreover, even though high level value function convergence analysis of the TD algorithms have been considered by a number of researchers, since the closed form solution of the value function in terms of the system dynamics has never been derived, value function parameter convergence to their respective correct value have not been shown. Thus, a novel approach for the value function learning analysis has not been delivered so far. In addition, the role of the value function discount factor selection and the type of the control signal in value function learning has not been examined in the literature.

In this PhD thesis, in order to fill these gaps in TD learning analysis, a low level analysis of the value learning by using a simple exemplar problem is performed which yields a closed form terminal value function solution consisting of the optimal parameters and optimal polynomial basis. Since the optimal solution is available for the value function, it is now possible to have a generic approach for explicit TD learning analysis.

The key result obtained from this analysis is the existence of the non-zero trajectory null space associated with the higher order parameter. As there is no information in the corresponding higher order basis, the higher order parameter cannot be learned/updated until the terminal sample is reached and as a result of this property, the higher parameter; hence the value function learning process is slow. The simulation results also confirm zero parameter update for this higher order parameter during the state trajectory. In addition, the simulation results reveal that the bias parameter starts learning first. It is also highlighted that this higher order parameter cannot be learned even at the final sample unless termination of the simulation is handled properly. To overcome this finite horizon effect, three generic approaches are presented.

A further contribution of the thesis is to address the role of the control signal on the value function learning. It is proved that the bang-bang control approach causes chaotic dynamics and this chaotic dynamics leads to digital offset and roughness in the chaotic region of the learned value function. To eliminate this problem, a dead-zone control approach is suggested.

In order to eliminate the chaotic dynamics and perform a realistic value function learning analysis, another test problem with saturated piecewise linear control is considered. It is shown that the non-zero trajectory null space exists for both higher order parameters up to the saturation threshold of the control signal. However, after the saturation threshold, these higher order parameters are learned as their bases gain information. This is another contribution given in the thesis.

A further contribution is the proof of the badly conditioned higher order parameters learning. Based on the value of the saturation threshold and the value function discount factor, badly conditioned learning occurs and to avoid this problem appropriate saturation threshold and discount factor value for this test problem are specified. A sound proof is provided for the rate of convergence analysis of the value function learning as well. It is shown that the residual gradient learning algorithm converges faster than the TD(0) algorithm for this test problem.

To generalize the insights gained from the novel parameter convergence and rate of convergence analysis in terms of utilizing the first order dynamics, the RL algorithm is applied to a higher order system. The learned value functions and control actions are compared with their respective optimal solutions and shown that the RL algorithm is able to learn the approximate solutions. The impact of non-zero trajectory null space and chaotic dynamics on the value function learning and control policies for higher order systems are addressed. This kind of attempt does not exist in the literature either.

Finally, CPG based RL algorithm is applied to a single robot leg by using the MapleSim multi-body modelling software and Maple/Modelica software properties. A symbolic IK solution is obtained by utilizing MapleSim advantages. The convergence results for the CPG and the RL results are presented and analysed. It is shown that the

CPG based RL algorithm is able to generate appropriate control signals for both joints of the leg which satisfy the learning targets.

As future work, value function parameter convergence and rate of convergence analysis will be performed when the test problem has random noise and also when the algorithm has random exploration noise. Hence, it will be possible to analyze the role of the noise in value function learning. Moreover, possible impacts of modelling error on the value function and control policy learning will be analyzed. In order to generalize the gained insights from the simple test problem based value function learning analyses, a test problem with higher order dynamics will be performed. The CPG based RL algorithm will initially be applied to a planer link robot and then a 3D robot. Finally, physical experiments will be performed.

Reference:

1. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *Artificial Neural Networks*, vol. SMC-13, pp. 81-93, Sept. 1983.
2. R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," Cambridge, The MIT Press, 1998, pp. 1-300.
3. R. S. Sutton, "Learning to predict by the methods of temporal differences. Machine Learning," *Machine Learning*, vol. 3, pp. 9-44, Aug. 1988.
4. D. P. Bertsekas, "Temporal difference methods for general projected equations," *IEEE Trans. on Aut. Cont.*, vol. 56, pp.1-12, Sep. 2011.
5. D. Kenji, "Reinforcement learning in continuous time and space," *Neural Computation* vol. 12, pp. 219-245, Jan. 2000.
6. F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circ. and Sys. Magazine*, vol. 9, pp.32-50, Sept. 2009.
7. X. L. Xu and D. C. Hu, "Continuous-action reinforcement learning with fast policy search and adaptive basis function selection," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 15, pp. 1055-1070, Mar. 2011.
8. H. R. Maei, C. Szepesvári, S. Bhatnagar, D. Precup, D. Silver and R. S. Sutton, "Convergent temporal-difference learning with arbitrary smooth function approximation," in *Advances in Neural Inf. Proc. Sys.*, 2009, pp. 1204-1212.
9. R. Legenstein, N. Wilbert and L. Wiskott, "Reinforcement learning on slow features of high dimensional input streams," *Plos. Comput. Biol.* Vol. 6, pp. 1-13, Aug. 2010.
10. C. Watkins, "Learning from delayed rewards," Ph.D dissertation, Psychology Dept. University of Cambridge, Cambridge, 1989.
11. W. B. Powell and M. Jun, (Sept. 2010). A review of stochastic algorithms with continuous value function approximation and some new approximate policy iteration algorithms for multi dimensional continuous application. Department of Operations Research and Financial Engineering, Princeton University, [Online]. Available: <http://www.castlelab.princeton.edu/Papers/PowellMa-JCTAsurvey%20paperSept152010%20.pdf>
12. P. Dayan and B.W. Balleine, "Reward, Motivation, and Reinforcement Learning," *Neuron*, vol. 36, pp. 285-298, Oct. 2002.13.

13. Fei-Yue, W., Z. Huaguang, and L. Derong, Adaptive Dynamic Programming: An Introduction. Computational Intelligence Magazine, IEEE, 2009. 4(2): p. 39-47.
14. L. Busoniu, D. Ernst, S. De and R. Babuska, "Approximate reinforcement learning: An overview," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium*, pp. 1-8.
15. M. Geist and O. Pietquin, "Parametric value function approximation: A unified view," in *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium*, pp. 9-16.
16. X. Xu, H. G. He and D. Hu, "Efficient reinforcement learning using recursive least-squares methods," *Journal of Artificial Intelligence Research* vol. 16, pp. 259-292, May, 2002.
17. R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9-44, Aug. 1988.
18. A. K. Agogino and K. Tumer, "Unifying temporal and structural credit assignment problems," *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, 2004, pp. 980-987.
19. J. N. Tsitsiklis, and B. V. Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. on Automatic Control*, vol. 42, pp. 674-690, May, 1997.
20. J. Z. Kolter, and A. Y. NG, "Regularization and feature selection in least-squares temporal difference learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 521-528.
21. A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210-229, July 1959.
23. G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, pp.257-277, May 1992.
24. B. Tanner and R. S. Sutton, "TD(λ) networks: temporal-difference networks with eligibility traces," in *Proc. of the 22nd Int. Conference on Machine Learning*, 2005, pp. 888-895.
25. H. Kimura and S. Kobayashi, "An analysis of actor/critic algorithms using eligibility traces: reinforcement learning with imperfect value function," in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 278-286.
26. S. P. Singh, R. S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, *Machine Learning*, vol. 22, pp. 123-158, Jan. 1996.

27. S. J. Bradke and A. J. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, pp. 33-57, Mar. 1996.
28. M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107-1149, Dec. 2003.
29. M. J. Mataric, "Reinforcement learning in the multi-robot domain in robot colonies," Verlag: Springer, 1997, pp. 73-83.
30. J. Schaeffer, M. Hlynka and V. Jussila, "Temporal difference learning applied to a high-performance game-playing program," in *Proc. of the 17th Int. Joint Conference on Artificial intelligence*, 2001, pp. 529-534.
31. L. P. Kaelbling, M.L. Littman and A.W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, May 1996.
32. S. Gadaleta and G. Dangelmayr, "Reinforcement learning chaos control using value sensitive vector-quantization," in *Neural Networks, 2001 Proc. Int.l Joint Con.*, pp. 996-1001.
33. L. J. Lin, "Programming robots using reinforcement learning and teaching," in *Proc. of the Ninth National Con. on Artificial Intelligence*, 1991, pp. 781-786.
34. H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robotics and Autonomous Systems*, vol. 22, pp. 283-302, Dec. 1996.
35. R. Tedrake, T. W. Zhang and H. S. Seung. "Stochastic policy gradient reinforcement learning on a simple 3D biped," in *Int. Con. on Intelligent Robots and Systems*, 2004, pp. 2849-2854.
36. G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi and G. Cheng, "Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot," *Int. Journal of Robotic Research*, vol. 27, pp. 213-228, Feb. 2008.
37. J. A. Boyan, "Least-Squares Temporal Difference Learning," *Machine Learning*, vol. 49, pp. 233-246, Nov. 2002.
38. J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: safely approximating the value function," in *Advance in Neural Information Processing Systems*, Cambridge, MA: The MIT, 1995, pp. 369-376.
39. J. N. Tisitsiklis and B. V. Roy, "Feature-based methods for large scale dynamic programming," in *Machine Learning*, Boston, Kluwer Academic Publishers, 1996, pp. 59-94.

40. L. J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," in *Machine Learning*, Boston, Kluwer Academic Publishers, 1992, pp. 293-321.
41. R. S. Sutton, "Generalization in reinforcement learning: successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems 8*, 1996, pp. 1038-1044.
42. R. S. Sutton, "Temporal credit assignment in reinforcement learning," Ph.D. Dissertation, Computer Sci. Dep., Univ. of Massachusetts, Amherst, 1982.
43. M. Tokic, "Adaptive ϵ -greedy exploration in reinforcement learning based on value differences," in *Proceedings of the 33rd Annual German Conference on Advances in Artificial Intelligence*, 2010, pp. 203-210.
44. M. Tokic and G. Palm, "Gradient algorithms for exploration/exploitation trade-offs: global and local variants" in *Proc. ANNPR'12 Proceedings of the 5th INNS IAPR TC 3 GIRPR Conference on Artificial Neural Networks in Pattern Recognition*, 2012, pp. 60-71.
45. S. B. Thrun, "Efficient exploration in reinforcement learning," Carnegie Mellon Univ., Pittsburgh, Tech. Rep. Jan. 1992.
46. O. Tutsoy and M. Brown. (2010, Sep.) "Convergence analysis of reinforcement learning approaches to humanoid locomotion," in *UK Automatic Control Conference (UKACC)*, 2010.
47. P. Dayan, "The convergence of TD(λ) for general λ ," *Machine Learning*, vol. 8, pp. 341-362, May, 1992.
48. R. Schapire, M. K. Warmuth and P. Kaelbling, "On the worst-case analysis of temporal-difference learning algorithms," *Machine Learning*, vol. 22, pp. 95-121, Mar. 1996.
49. V. Tadić, "On the convergence of temporal-difference learning with linear function approximation," *Machine Learning*, vol. 42, pp. 241-267, Mar. 2001.
50. R.S. Sutton, C Szepesvari and H. R. Maei, "A convergent O(n) temporal-difference algorithm for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems 21*, 2009, pp.1-8.
51. R. Schoknecht and A. Merke, "TD(0) Converges Provably Faster than the Residual Gradient Algorithm," in *Proc. of the Twentieth Int. Conf. on Machine Learning*, 2003, pp.680-687.
52. L. Li, "A worst-case comparison between temporal difference and residual gradient with linear function approximation," in *Proc. of the 25th Int. Conf. on Machine Learning*, 2008, pp. 560-567.

53. D. D. Dier and S. S. Joshi, "Reinforcement learning solution to a benchmark time optimal control problem," in *Navigation and Control Conference and Exhibit*, 2007.
54. F. L. Lewis and V.L. Syrmos, "*Optimal control*," 2nd ed., New York: Wiley, 1995, pp. 10-540.
55. L. Busoniu, R. Babuska., B. D. Schutter and D. Ernst, "Reinforcement Learning and Dynamic Programming Using Function Approximators," 1st ed., Boca Raton, 2010, pp. 1-270.
56. A. A. Sherstov and P. Stone, "Function Approximation via Tile Coding: Automating Parameter Choice," in *In Proc. Sym. on Abstraction, Reformulation, and Approximation (SARA-05)*,". 2005, pp. 901-901.
57. J. S. R. Jang, C. T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. on Neural Networks*, vol. 4, pp. 156-159, Jan. 1993.
58. L. Baird, "Residual algorithms: reinforcement learning with function approximation," in *In Proc. of the Twelfth Int. Con. on Machine Learning*, 1995, pp. 30-37.
59. Y. Nakamura, T. Mori, M. A. Sato and S. Ishii, "Reinforcement learning for a biped robot based on a CPG-actor-critic method," *Neural Networks*, vol. 20, pp. 723-735, Aug. 2007.
60. Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Nigaki, and K. Fujimura, "The intelligent ASIMO: system overview and integration.," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 2002, pp. 2478-2483.
61. R. Sellaouti, O. Stasse, S. Kajita, K. Yokoi and A. Kheddar, "Faster and Smoother Walking of Humanoid HRP-2 with Passive Toe Joints," in *Proc. of the 2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006, pp. 4909-4914.
62. S. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "Humanoid Robots in Waseda University—Hadaly-2 and WABIAN," *Autonomous Robots*, vol. 12, pp. 25-38, Jan. 2002.
63. G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori, "The iCub humanoid robot: an open platform for research in embodied cognition," in *Proc. of the 8th Workshops on Performance Metrics for Intelligent Systems*, 2008, pp. 50-56.
64. J. Peters, S. Vijayakumar and S. Schaal, "Reinforcement learning for humanoid robotics," in *Third IEEE-RAS Int.Conf. on Humanoid Robots*, 2004, pp. 1-17.

65. E. A. C. Filho, Humanoid Robots: New Developments, Advanced Robotic Systems International I-Tech. [Online]. Available: <http://dc172.4shared.com/doc/O2nCmV2V/preview.html>.
66. F. Kanehiro, K. Fujiwara, S. Kajita, K. Yokoi, K. Kaneko, H. Hirukawa, Y. Nakamura and K. Yamane, "Open architecture humanoid robotics platform," in *Proceedings. ICRA '02. IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 24-30.
67. S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita and H. Hirukawa, "Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2007, pp. 3641-3647.
68. G. A. Medrano-Cerda, H. Dallali, M. Brown, N. G. Sagarakis, D. G. Caldwell, "Modelling and simulation of the locomotion of humanoid robots, in *UK Automatic Control Conference (UKACC)*, 2010.
69. H. Dallali, "Modelling and dynamic stabilisation of a compliant humanoid robot, CoMan," Ph.D dissertation, *Electrical and Electronic Eng.* The Univ. of Manchester, Manchester, 2012.
70. P. Goessens and T. Richard, (2011, Nov.). Case Study: Learning toWalk Faster with High Performance Modelling, MapleSim Coop. [online].
71. Z. Cai, R. Lazarov, T. A. Manteuffel and S. F. McCormick, "First-order system least squares for second-order partial differential equations," *SIAM Journal on Numerical Analysis*, vol. 34, pp. 425-454, Dec. 1997.
72. P. Glendinning and P. Kowalczyk, "Micro-chaotic dynamics due to digital sampling in hybrid systems of Filippov type," *Physica*, vol. 239, pp. 58-71, Jan. 2010.
73. P. Glendinning, "Stability, instability, and chaos : an introduction to the theory of nonlinear differential equations," New York: Cambridge University Press, 1994, pp. 1-388.
74. G. Kreisselmeier, B. D. O. Anderson, "Robust model reference adaptive control," *IEEE Trans. on Automatic Control*, vol. 31, pp. 127-133, Feb. 1986.
75. M. Brown and O. Tutsoy, "On the value function convergence analysis of temporal difference algorithms with bang-bang control," *IEEE Trans. on Neural Networks and Learning Systems*, submitted.
76. M. Brown, and O. Tutsoy, "On the analysis of parameter convergence for temporal difference learning of an exemplar balance problem," in *Proc. of the 12th An. Con. on Towards Autonomous Robotic Systems*, 2011, pp. 404-405.
77. M. Brown and O. Tutsoy, "An exemplar test problem on parameter convergence analysis of temporal difference algorithms," in *10th World*

Congress on Intelligent Control and Automation (WCICA), 2012, pp. 2925-2930.

78. O. Tutsoy and M.Brown “An analysis of value function learning with piecewise linear control,” *IEEE Trans. on Neural Networks and Learning Systems*, submitted.
79. O. Tutsoy, M.Brown, H. Wang, “Convergence analysis of temporal difference learning algorithms based on a test problem,” in *The 14th International Conference on Control and Applications (IASTED)*, 2012.
80. O. Tutsoy, M. Brown and H. Wang, “Reinforcement learning algorithm application and multi-body system design by using MapleSim and Modelica,” in *International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2012, pp. 650-655.
81. G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi and G. Cheng, “Learning CPG-based biped locomotion with a policy gradient method: application to a humanoid robot,” vol. 27, pp. 213-228, Feb. 2008.
82. A. M. Arsenio, “ Tuning of neural oscillators for the design of rhythmic motions,” in *Proc. of the IEEE Int. Conference on Robotics and Automation*, 2000, pp. 1888-1893.
83. S. R. Buss, (2009, Oct.), “Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods,” Dept. of Math. Uni. of California, San Diego. [Online]. Available: <http://math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf>.
84. M. Raison, “On the quantification of joint and muscle efforts in the human body during motion,” Ph.D. dissertation, Electrical Eng. Université Catholique de Louvain, Louvain-la-Neuve, 2009.
85. P. Goossens and T. Richard, (2011, Nov.) Using symbolic technology to derive inverse kinematic solutions for actuator control development. Maplesoft, Waterloo, Canada. [Online]. Available: http://www.maplesoft.com/Whitepapers/Mathmod2012_pgoossens_trichard_p_reprint_paper.pdf
86. O. Tutsoy, M.Brown, “CPG based RL algorithm learns to control of a humanoid robot leg,” *IEEE Trans. on Robotics*, submitted.

Appendix A

Closed form solution for the first order system used in the chapters 3 and 4 can be obtained as:

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau \quad (94)$$

where A and B are the continuous time parameters and u is the control signal which is constant for the test problem analysis; hence dependence on τ is dropped. Denote that $t = hk$ and re-express the equation (94) as:

$$x_k = e^{Ahk}x_0 + \left(\int_0^{hk} e^{A(hk-\tau)} d\tau \right) Bu_k \quad (95)$$

With the solution of the integral part, the equation (95) becomes:

$$x_k = e^{Ahk}x_0 + \left(A^{-1}(e^{Ahk} - I) \right) Bu_k \quad (96)$$

Now denote $e^{Ah} = a$, $\left(A^{-1}(e^{Ah} - I) \right) B = b$ and $u_k = -\text{sign}(x_0)$, then the equation (96) is:

$$x_k = a^k x_0 + (a^k - 1) \cdot \frac{b}{a - 1} (-\text{sign}(x_0)) \quad (97)$$

where A is assumed to be unity for this analysis. Substitute $\text{sign}(x_0) = s(x_0)$ for the sake of simplicity and re-organizing the expression (97) gives:

$$x_k = a^k \left(x_0 - \frac{b}{a - 1} s(x_0) \right) + \frac{b}{a - 1} s(x_0) \quad (98)$$

This is the closed form solution for the first order system used as an exemplar problem for the value function learning analysis in the chapters 3 and 4. The next section will explicitly derive the closed form terminal finite horizon value function solution based on this closed form solution.

Finite Horizon Sample

Initially determine the sample, which is basically finite horizon sample, where the closed form dynamics of the first order system is zero. Therefore, the equation (98) can be written as:

$$0 = a^k \left(x_0 - \frac{b}{a-1} s(x_0) \right) + \frac{b}{a-1} s(x_0) \quad (99)$$

Call $k = k^*$ when the terminal dynamics of the system is zero. Re-organizing the equation (99) yields:

$$-\frac{b}{a-1} s(x_0) = a^{k^*} \left(x_0 - \frac{b}{a-1} s(x_0) \right) \quad (100)$$

Taking natural log of the both side and expanding the right hand side of the equation (100) leads:

$$\ln \left(-\frac{b}{a-1} s(x_0) \right) = k^* \ln(a) + \ln \left(x_0 - \frac{b}{a-1} s(x_0) \right) \quad (101)$$

Take the term k^* to the right and all the other terms are on the left.

$$k^* \ln(a) = \ln \left(-\frac{b}{a-1} s(x_0) \right) - \ln \left(x_0 - \frac{b}{a-1} s(x_0) \right) \quad (102)$$

Hence the general solution for the finite horizon sample k^* is:

$$k^* = \left[\ln \left| -\frac{b}{a-1} s(x_0) \right| - \ln \left| x_0 - \frac{b}{a-1} s(x_0) \right| \right] / \ln(a) \quad (103)$$

This finite horizon expression is valid for both negative and positive initial states.

Closed Form Terminal Value Function Solution with Bang-Bang Control

In this section in terms of using the finite horizon expression given by the equation (103) and the closed form solution for the system introduced by the equation (98), the closed form terminal value function solution will be derived which will be used for the parameter convergence and rate of convergence analysis.

$$V(x_0) = \sum_{t=0}^{k^*} \gamma^t \left| \frac{b}{a-1} s(x_0) + a^t \left(x_0 - \frac{b}{a-1} s(x_0) \right) \right| \quad (104)$$

Initially assume that the initial state is negative, then (104) becomes:

$$V(x_0) = \underbrace{-\frac{b}{a-1} s(x_0) \sum_{t=0}^{k^*} \gamma^t}_I - \underbrace{\left(x_0 - \frac{b}{a-1} s(x_0) \right) \sum_{t=0}^{k^*} (\gamma a)^t}_II \quad (105)$$

The I part solution of the equation (105) is:

$$\begin{aligned} I &= -\frac{b}{a-1} s(x_0) \frac{1-\gamma^{k^*+1}}{1-\gamma} \\ &= -\frac{b}{a-1} s(x_0) \frac{1}{1-\gamma} + \frac{b}{a-1} s(x_0) \frac{\gamma}{1-\gamma} \gamma^{k^*} \\ &= -\frac{b}{a-1} s(x_0) \frac{1}{1-\gamma} + \frac{b}{a-1} s(x_0) \frac{\gamma}{1-\gamma} \left(e^{\ln\left(-\frac{b}{a-1} s(x_0)\right)} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(e^{\ln\left(x_0 - \frac{b}{a-1} s(x_0)\right)} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \end{aligned} \quad (106)$$

The final expression for the equation (106) is:

$$I = -\frac{b}{a-1} s(x_0) \frac{1}{1-\gamma} + \frac{b}{a-1} s(x_0) \frac{\gamma}{1-\gamma} \left(-\frac{b}{a-1} s(x_0) \right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(x_0 - \frac{b}{a-1} s(x_0) \right)^{\frac{\ln(\gamma)}{\ln(a)}} \quad (107)$$

The second part solution of the equation (105) is given by:

$$\begin{aligned} II &= -\left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{1-(\gamma a)^{k^*+1}}{1-(\gamma a)} \\ &= -\left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{1}{1-\gamma a} + \left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{\gamma a}{1-\gamma a} (\gamma a)^{k^*} \\ &= -\left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{1}{1-\gamma a} + \left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{\gamma a}{1-\gamma a} \left(e^{\ln\left(-\frac{b}{a-1} s(x_0)\right)} \right)^{\frac{\ln(\gamma a)}{\ln(a)}} \left(e^{\ln\left(x_0 - \frac{b}{a-1} s(x_0)\right)} \right)^{\frac{\ln(\gamma a)}{\ln(a)}} \end{aligned} \quad (108)$$

Re-organizing the equation (108) gives:

$$\begin{aligned} II &= -\left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{1}{1-\gamma a} + \left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{\gamma a}{1-\gamma a} \left(-\frac{b}{a-1} s(x_0) \right)^{\frac{\ln(\gamma a)}{\ln(a)}} \left(x_0 - \frac{b}{a-1} s(x_0) \right)^{\frac{\ln(\gamma a)}{\ln(a)}} \\ &= -\left(x_0 - \frac{b}{a-1} s(x_0) \right) \frac{1}{1-\gamma a} + \left(x_0 - \frac{b}{a-1} s(x_0) \right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(-\frac{b}{a-1} s(x_0) \right)^{\frac{\ln(\gamma a)}{\ln(a)}} \frac{\gamma a}{1-\gamma a} \end{aligned} \quad (109)$$

Summing up the first and second parts of the terminal value function solution gives the final closed form solution for the test problem.

$$V(x_0) = -\frac{b}{a-1}s(x_0)\frac{1}{1-\gamma} - \left(x_0 - \frac{b}{a-1}s(x_0)\right)\frac{1}{1-\gamma a} + \dots$$

$$\dots \left(x_0 - \frac{b}{a-1}s(x_0)\right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(-\frac{b}{a-1}s(x_0)\right)^{\frac{\ln(\gamma a)}{\ln(a)}} \left(\frac{\gamma a}{1-\gamma a} - \frac{\gamma}{1-\gamma}\right) \quad (110)$$

The generalized version of the solution given by (110) is:

$$V(x_k) = \left|-\frac{b}{a-1}s(x_k)\right|\frac{1}{1-\gamma} - \left|x_k - \frac{b}{a-1}s(x_k)\right|\frac{1}{1-\gamma a} + \dots$$

$$\dots \left|x_k - \frac{b}{a-1}s(x_k)\right|^{-\ln(\gamma)/\ln(a)} \left|-\frac{b}{a-1}s(x_k)\right|^{\ln(\gamma a)/\ln(a)} \left(\frac{\gamma a}{1-\gamma a} - \frac{\gamma}{1-\gamma}\right) \quad (111)$$

This solution is used for the parameter convergence and rate of convergence analysis in order to explain the learning properties of the TD(0) and residual gradient learning algorithms.

Appendix B

In this part of the thesis, closed form terminal value function based on a test problem addressed in the chapter 4 will be derived step by step. The key difference between the chapters 3 and 4 is the control part. In this analysis a saturated piecewise linear control, which is the less smooth version of the $\tanh()$ function, is considered. A further difference is that the previous value function derivation is based on finite horizon whereas in this section the value function derivation is infinite horizon.

The Sample at the Saturation Threshold

Initially let define the sample to reach the saturation threshold of the control signal. The sample k^* is the sample where the state $x_k = c$ which is basically the saturation threshold.

$$x_{k^*} = -c = a^{k^*} \left(x_0 - \frac{b}{a-1}s(x_0)\right) + \frac{b}{a-1}s(x_0) \quad (112)$$

For the simplicity, substitute $w_0 = \frac{b}{a-1}s(x_0)$ and re-arrange the equation (112).

$$-w_0 - c = a^{k^*} (x_0 - w_0) \quad (113)$$

Taking the log of both sides yields:

$$\ln(-w_0 - c) = k^* \ln(a) + \ln(x_0 - w_0) \quad (114)$$

Therefore, the sample to reach the saturation threshold is given by:

$$k^* = \ln(-w_0 - c) / \ln(a) - \ln(x_0 - w_0) / \ln(a) \quad (115)$$

The general form of the equation (115) is:

$$k^* = \ln(|w_0| - c) / \ln(a) - \ln|x_0 - w_0| / \ln(a) \quad (116)$$

Now in terms of using this sample at the saturation threshold of the control signal, the value function solution is obtained next up $k^* + 1$ sample.

The Value Function Solution up to Saturation Threshold

In the chapter 4, the instantaneous reward is quadratic both in state and control signal.

$$r(x_k, u_k) = qx_k^2 + \lambda u_k^2 \quad (117)$$

The quadratic form of the closed form solution for the test problem is:

$$x_k^2 = w_0^2 + 2a^k w_0 (x_0 - w_0) + a^{2k} (x_0 - w_0)^2 \quad (118)$$

The control signal is unity up to the saturation threshold. Thus, the value function expression up to the saturation threshold is given by:

$$\begin{aligned} V^1 = & \underbrace{q \sum_{t=0}^{k^*} \gamma^t w_0^2}_{Ia} + \underbrace{q \sum_{t=0}^{k^*} (\gamma a)^t w_0 (x_0 - w_0)}_{Ib} + \dots \\ & \dots \underbrace{q \sum_{t=0}^{k^*} (\gamma a^2)^t (x_0 - w_0^2)}_{Ic} + \underbrace{\lambda \sum_{t=0}^{k^*} \gamma^t (-s(x_0)^2)}_{Ic} \end{aligned} \quad (119)$$

The solution for the Ia part of the equation (119) is:

$$\begin{aligned}
Ia &= q w_0^2 \sum_{t=0}^{k^*} \gamma^t \Rightarrow q w_0^2 \frac{1 - \gamma^{k^*+1}}{1 - \gamma} \\
&= w_0^2 \frac{q}{1 - \gamma} - w_0^2 \frac{q \gamma}{1 - \gamma} \gamma^{k^*+1} \\
&= w_0^2 \frac{q}{1 - \gamma} - w_0^2 \frac{q \gamma}{1 - \gamma} \left(e^{\ln(w_{0,c})} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(e^{\ln|x_0 - w_0|} \right)^{-\frac{\ln(\gamma)}{\ln(a)}} \\
&= w_0^2 \frac{q}{1 - \gamma} - w_0^2 \frac{q \gamma}{1 - \gamma} (w_{0,c})^{\frac{\ln(\gamma)}{\ln(a)}} (|x_0 - w_0|)^{-\frac{\ln(\gamma)}{\ln(a)}}
\end{aligned} \tag{120}$$

where $w_{0,c} = |w_0| - c$. The solution for *Ib* part of the equation (119) is given by:

$$\begin{aligned}
Ib &= 2q w_0 (x_0 - w_0) \sum_{t=0}^{k^*} (\gamma a)^t \Rightarrow -2q |x_0 - w_0| \frac{1 - (\gamma a)^{k^*+1}}{1 - \gamma a} \\
&= -|x_0 - w_0| \frac{2q}{1 - \gamma a} + |x_0 - w_0| \frac{2q \gamma a}{1 - \gamma a} (\gamma a)^{k^*+1} \\
&= -|x_0 - w_0| \frac{2q}{1 - \gamma a} + |x_0 - w_0| \frac{2q \gamma a}{1 - \gamma a} \left(e^{\ln(w_{0,c})} \right)^{\frac{\ln(\gamma a)}{\ln(a)}} \left(e^{\ln|x_0 - w_0|} \right)^{-\frac{\ln(\gamma a)}{\ln(a)}} \\
&= -|x_0 - w_0| \frac{2q}{1 - \gamma a} + \frac{2q \gamma a}{1 - \gamma a} (w_{0,c})^{\frac{\ln(\gamma a)}{\ln(a)}} |x_0 - w_0|^{-\frac{\ln(\gamma a)}{\ln(a)}}
\end{aligned} \tag{121}$$

The solution for the *Ic* of the equation (119) is:

$$\begin{aligned}
Ic &= q (x_0 - w_0)^2 \sum_{t=0}^{k^*} (\gamma a^2)^t \Rightarrow q (x_0 - w_0)^2 \frac{1 - (\gamma a^2)^{k^*+1}}{1 - \gamma a^2} \\
&= (x_0 - w_0)^2 \frac{q}{1 - \gamma a^2} - (x_0 - w_0)^2 \frac{q \gamma a^2}{1 - \gamma a^2} (\gamma a^2)^{k^*+1} \\
&= (x_0 - w_0)^2 \frac{q}{1 - \gamma a^2} - (x_0 - w_0)^2 \frac{q \gamma a^2}{1 - \gamma a^2} \left(e^{\ln(w_{0,c})} \right)^{\frac{\ln(\gamma a^2)}{\ln(a)}} \left(e^{\ln|x_0 - w_0|} \right)^{-\frac{\ln(\gamma a^2)}{\ln(a)}} \\
&= (x_0 - w_0)^2 \frac{q}{1 - \gamma a^2} - \frac{q \gamma a^2}{1 - \gamma a^2} (w_{0,c})^{\frac{\ln(\gamma a^2)}{\ln(a)}} |x_0 - w_0|^{-\frac{\ln(\gamma a^2)}{\ln(a)}}
\end{aligned} \tag{122}$$

The solution for the *Id* part is given as:

$$\begin{aligned}
Id &= \lambda s(x_0)^2 \sum_{t=0}^{k^*} \gamma^t \Rightarrow \lambda s(x_0)^2 \frac{1-\gamma^{k^*+1}}{1-\gamma} \\
&= \lambda s(x_0)^2 \frac{1}{1-\gamma} - \lambda s(x_0)^2 \frac{\gamma}{1-\gamma} \gamma^{k^*+1} \\
&= \lambda s(x_0)^2 \frac{1}{1-\gamma} - \lambda s(x_0)^2 \frac{\gamma}{1-\gamma} \left(e^{\ln(w_{0,c})} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(e^{\ln|x_0-w_0|} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \\
&= \lambda s(x_0)^2 \frac{1}{1-\gamma} - \lambda s(x_0)^2 \frac{\gamma}{1-\gamma} (w_{0,c})^{\frac{\ln(\gamma)}{\ln(a)}} |x_0 - w_0|^{\frac{\ln(\gamma)}{\ln(a)}}
\end{aligned} \tag{123}$$

Finally, summation of the all solutions yields the terminal value function solution up to the saturation threshold.

$$\begin{aligned}
V^1 &= w_0^2 \left(\frac{q}{1-\gamma} + \frac{\lambda s(x_0)^2}{(1-\gamma)w_0^2} \right) - |x_0 - w_0| q \frac{2}{1-\gamma a} + (x_0 - w_0)^2 q \frac{1}{1-\gamma a^2} \\
&\quad + |x_0 - w_0|^{\frac{\ln(\gamma)}{\ln(a)}} \left(w_{0,c}^{\frac{\ln(\gamma a)}{\ln(a)}} \frac{2\gamma q a}{1-\gamma a} - w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \frac{\gamma (q w_0^2 + \lambda s(x_0)^2)}{1-\gamma} - w_{0,c}^{\frac{\ln(\gamma a^2)}{\ln(a)}} \frac{\gamma q a^2}{1-\gamma a^2} \right)
\end{aligned} \tag{124}$$

The Value Function Solution for the Linear Part of the Control Signal

In this section, the second part of the value function which corresponds to the linear component of the saturated piecewise linear control will be derived. The second part of the solution will start from k^*+1 sample and the terminal sample will be ∞ . However, as the value function representation has a discount factor, based on the value of the discount factor, the value function will act like finite horizon. Now let derive the closed form solution of the system with the linear control $u_k = -x_k/c$.

$$x_{k+1} = ax_k + b(-x_k/c) \tag{125}$$

This can be re-organized as:

$$x_{k+1} = (a - b/c)x_k \tag{126}$$

Then the closed form solution for the system is:

$$x_k = (a - b/c)^k x_0 \tag{127}$$

For the sake of simplicity $a(c) = (a - b/c)$ is denoted. The value function is again quadratic in the state and the control signal.

$$V'' = q \sum_{t=k^*+1}^{\infty} \gamma^t \left(a(c)^2 \right)^t x_t^2 + \lambda \sum_{t=k^*+1}^{\infty} \gamma^t \left(a(c)/c \right)^{2t} x_t^2 \quad (128)$$

where V'' represents the terminal value function solution for the linear part of the saturated piecewise linear control. This expression can be written as:

$$V'' = \underbrace{qx_{k^*+1}^2 \gamma^{k^*+1} \sum_{t=0}^{\infty} \left(\gamma a(c)^2 \right)^t}_{2a} + \underbrace{\lambda \frac{x_{k^*+1}^2}{c^2} \gamma^{k^*+1} \sum_{t=0}^{\infty} \left(\gamma a(c)^2 \right)^t}_{2b} \quad (129)$$

The solution for the $2a$ part of the equation (129) is given by:

$$\begin{aligned} 2a &= qx_{k^*+1}^2 \gamma^{k^*+1} \sum_{t=0}^{\infty} \left(\gamma a(c)^2 \right)^t \\ &= qx_{k^*+1}^2 \gamma^{k^*} \frac{1}{1 - \gamma a(c)^2} \\ &= qx_{k^*+1}^2 \left(e^{w_{0,c}} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(e^{|x_0 - w_0|} \right)^{\frac{-\ln(\gamma)}{\ln(a)}} \frac{\gamma}{1 - \gamma a(c)^2} \\ &= x_{k^*+1}^2 \left(w_{0,c} \right)^{\frac{\ln(\gamma)}{\ln(a)}} |x_0 - w_0|^{\frac{-\ln(\gamma)}{\ln(a)}} \frac{q\gamma}{1 - \gamma a(c)^2} \end{aligned} \quad (130)$$

The solution for the $2b$ part of the equation (129) is:

$$\begin{aligned} 2b &= \lambda x_{k^*+1}^2 \gamma^{k^*+1} \sum_{t=0}^{\infty} \left(\gamma (a(c)/c)^2 \right)^t \\ &= \lambda \frac{x_{k^*+1}^2}{c^2} \gamma^{k^*} \frac{1}{1 - \gamma a(c)^2} \\ &= \lambda \frac{x_{k^*+1}^2}{c^2} \left(e^{w_{0,c}} \right)^{\frac{\ln(\gamma)}{\ln(a)}} \left(e^{|x_0 - w_0|} \right)^{\frac{-\ln(\gamma)}{\ln(a)}} \frac{\gamma}{1 - \gamma a(c)^2} \\ &= \frac{x_{k^*+1}^2}{c^2} \left(w_{0,c} \right)^{\frac{\ln(\gamma)}{\ln(a)}} |x_0 - w_0|^{\frac{-\ln(\gamma)}{\ln(a)}} \frac{\lambda\gamma}{1 - \gamma a(c)^2} \end{aligned} \quad (131)$$

The summation of the $2a$ and $2b$ gives the closed form solution of the value function for the linear part of the saturated piecewise linear control.

$$V^* = x_{k^*+1}^2 |x_0 - w_0|^{-\frac{\ln(\gamma)}{\ln(a)}} w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \left(\frac{q\gamma}{1-\gamma a(c)^2} + \frac{\lambda\gamma}{c^2(1-\gamma a(c)^2)} \right) \quad (132)$$

Thus, the whole closed form terminal value function solution for the test problem with the saturated piecewise linear control is:

$$\begin{aligned} V(x_k) = & w_0^2 \left(\frac{q}{1-\gamma} + \frac{\lambda s(x_0)^2}{(1-\gamma)w_0^2} \right) - |x_0 - w_0| q \frac{2}{1-\gamma a} + (x_0 - w_0)^2 q \frac{1}{1-\gamma a^2} \\ & + |x_0 - w_0|^{-\frac{\ln(\gamma)}{\ln(a)}} \left(w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \frac{2\gamma q a}{1-\gamma a} - w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \frac{\gamma (q w_0^2 + \lambda s(x_0)^2)}{1-\gamma} - w_{0,c}^{\frac{\ln(\gamma a^2)}{\ln(a)}} \frac{\gamma q a^2}{1-\gamma a^2} \right) \\ & + x_{k^*+1}^2 |x_0 - w_0|^{-\frac{\ln(\gamma)}{\ln(a)}} w_{0,c}^{\frac{\ln(\gamma)}{\ln(a)}} \left(\frac{q\gamma}{1-\gamma a(c)^2} + \frac{\lambda\gamma}{c^2(1-\gamma a(c)^2)} \right) \end{aligned} \quad (133)$$

The chapter 4 uses this solution for the parameter convergence and rate of convergence analysis. That chapter also analyses the role of the control signal on the value function learning.

Appendix C

In this section, instantaneous reward or known as the utility function, which is the measure of the one step of the cost, is formed in terms of the state and corresponding control signal of the system. Initially, discounted value function with the LQR cost will be derived and a simulation will be performed in order to show how the dynamics and control signal is behaving.

LQR Feedback Control Strategy

First of all the quadratic cost function in terms of both state and control action will be defined. Since it is an LQR control problem, corresponding Riccati equation must be solved to determine the gain of the controller. The LQR cost is:

$$J(x_k, u_k) = x_{k+t}^T Q x_{k+t}^t + u_{k+t}^T R u_{k+t} \quad (134)$$

Q and R to be positive definite so that the costs function is well defined. To solve the equation (134), apply Bellman's Optimality Principle and divide the sum into two.

$$J(x_k, u_k) = J(x_k, u_k) + \sum_{i=k+1}^{\infty} \gamma^{i-(k+1)} J(x_i, u_i) \quad (135)$$

The optimal value for the LQR is quadratic in the current state so that:

$$J(x_k, u_k) = x_k^T P x_k \quad (136)$$

where P is an unknown matrix that will be determined. Thus the Bellman equation given by (135) takes the form for the LQR,

$$x_k^T P x_k = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1} \quad (137)$$

Substitute $u_k = -Kx_k$ where K is the feedback gain which is to be determined and $x_{k+1} = Ax_k + Bu_k$. The resulting matrix is linear in P .

$$x_k^T P x_k = x_k^T \left(Q + K^T R K + (A - BK)^T P (A - BK) \right) x_k \quad (138)$$

Take the derivative of $J(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k + x_{k+1}^T P x_{k+1}$ with respect to u_k to obtain the minimum value of the value function in terms of control action.

$$\frac{\partial J(x_k, u_k)}{\partial u_k} = 2u_k^T R + 2B^T P (Ax_k + Bu_k) = 0 \quad (139)$$

$$u_k^* = -\left(R + B^T P B \right)^{-1} B^T P A x_k \quad (140)$$

Hence, K optimal feedback gain is defined as ($u_k^* = -Kx_k$),

$$K = \left(R + B^T P B \right)^{-1} B^T P A \quad (141)$$

Solving the Riccati equation given by (138) is difficult; hence an iteration method proposed by Hwer is considered [6].

$$K_{j+1} = \left(R + B^T P_{j+1} B \right)^{-1} B^T P_{j+1} A \quad (142)$$

Using the equation (142) and rearranging the equation (138) yields the following solution.

$$P_{j+1} = Q + A^T P_j A - A^T P_j B \left(R + B^T P_j B \right)^{-1} B^T P_j A \quad (143)$$

This iterative approach is valid under two assumptions. The first one is detectability; a system is called detectable if only if all of its unobservable states are stable. The

second assumption is stabilizability; a system is stabilizable if all uncontrollable states have stable dynamics.

LQR Control Simulation Results

A simulation is designed for a damped pendulum model and it is shown that the learned control policy is capable to take the states to the target position. The dynamics are given in linear form by $ml\ddot{x} = -\mu\dot{x} + mglx + u$ and the physical parameters are $m = l = 1$, $g = 9.8$, $\mu = 0.01$, mass, length, gravity and inertia of the pendulum. The discrete dynamics with 0.01 sampling time is:

$$x_{k+1} = \begin{pmatrix} 0.995 & 0.0998 \\ -0.0998 & 0.995 \end{pmatrix} x_k + \begin{pmatrix} 0.005 \\ 0.0998 \end{pmatrix} u_k \quad (144)$$

Now let see how the dynamics of the damped inverted pendulum dynamics and control signal evolve with the LQR control.

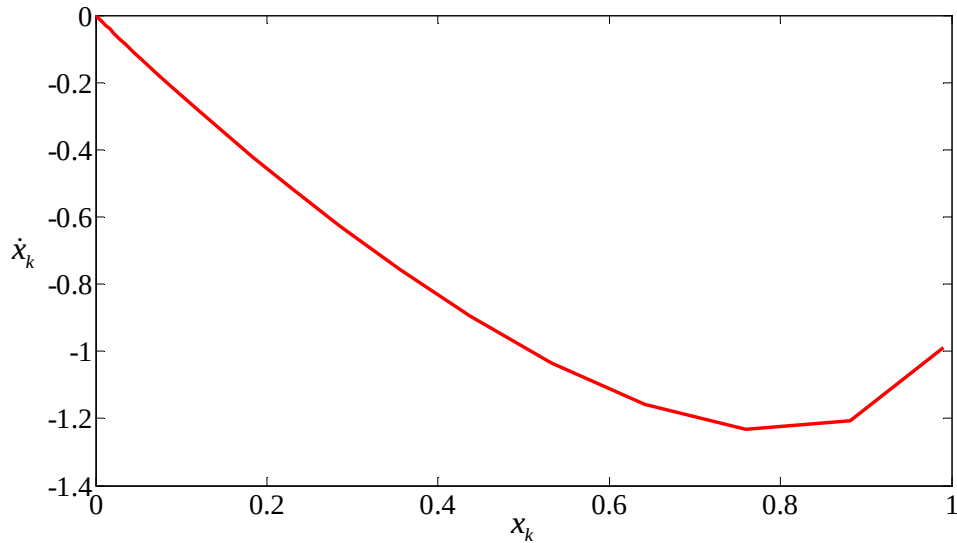


Figure 57: Dynamics of the damped inverted pendulum with LQR controller.

The length of the simulation is 100 to obtain P and K parameters specified by the equations (143) and (142). These values for the damped pendulum are $P = [19.6425 \ 4.1563; \ 4.1563 \ 14.0522]$ and $K = [0.3219 \ 1.2819]$. $R = 1$ and $Q = [1 \ 0; \ 0 \ 1]$. The length of the simulation for the dynamics of the pendulum shown in the Figure 57 is 250 with initial states $[0.99 \ -0.99]$. The feedback

controller gain is able to stabilize the system from its initial state. In addition, it is expected that the control action generated by the LQR feedback controller becomes smaller as the states shrink.

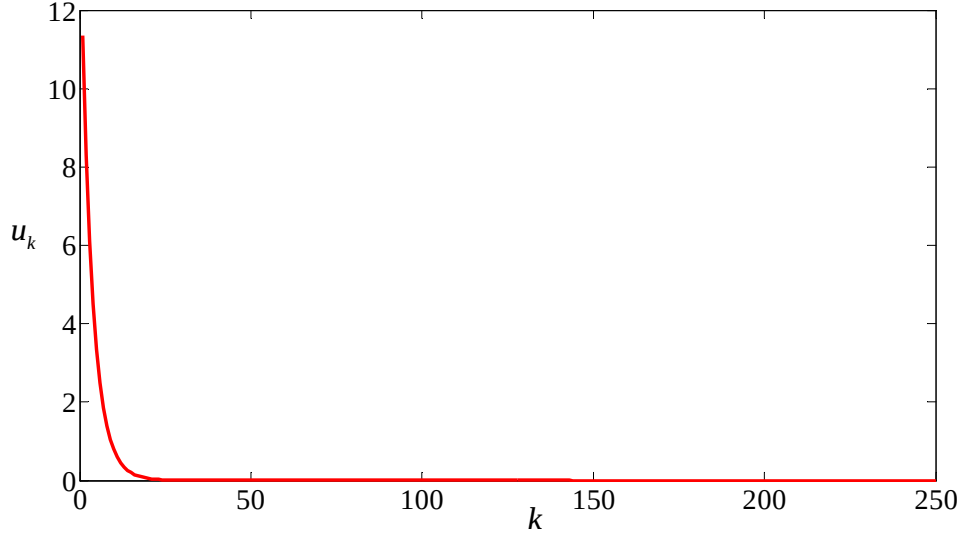


Figure 58: Control signal of LQR controller for the damped inverted pendulum.

Since the controller gain is fixed, the value of the control signal changes based on the states of the pendulum. As can be seen from the Figure 58, initially generated control signal is large and negative; however as the states evolve to the $[0 \ 0]$ final points, it becomes zero. Around sample 45, there is an overshoot which occurred during the stabilization around zero states.

Time Optimal Control Design for an Inverted Pendulum

In this section, initially time optimal/bang-bang control design of a plant with one positive real pole and one negative real pole, which is basically linear inverted pendulum model, will be introduced and then corresponding value function will be presented. Here, time optimal control of a plant having one pole on the positive real axis and another pole on the negative real axis is introduced. The state space model of the plant is:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 9.81 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u \quad (145)$$

where x_1' is the position state, x_2' is the velocity state of the model and u is the control signal that will be designed. Equation (145) is basically $\dot{x}' = Ax' + Bu$. The eigenvalues of the matrix A are basically poles of the plant. Let define Λ as a 2×2 diagonal matrix and its diagonals are eigenvalues of the plant.

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \quad (146)$$

For this particular example $\lambda_1 = 3.1321$ is the unstable real eigenvalue and $\lambda_2 = -3.1321$ is the real negative eigenvalue. Therefore, the plant is unstable. Then, there also exist a nonsingular matrix P which satisfies;

$$P^{-1}AP = \Lambda \quad (147)$$

The P matrix and inverse of P are,

$$P = \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix}, \quad P^{-1} = \frac{1}{\lambda_2 - \lambda_1} \begin{pmatrix} \lambda_2 & -1 \\ -\lambda_1 & 1 \end{pmatrix} \quad (148)$$

Now it is ready to write the plant dynamics in the modal form.

$$\begin{aligned} \dot{x}' &= \Lambda x' + P^{-1} \tilde{u} \\ \begin{pmatrix} \dot{x}_1' \\ \dot{x}_2' \end{pmatrix} &= \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} x_1' \\ x_2' \end{pmatrix} + \frac{1}{\lambda_2 - \lambda_1} \begin{pmatrix} \lambda_2 & -1 \\ -\lambda_1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ u \end{pmatrix} \end{aligned} \quad (149)$$

This can be written explicitly as below.

$$\dot{x}_1' = \lambda_1 x_1' - \frac{u}{\lambda_2 - \lambda_1}, \quad \dot{x}_2' = \lambda_2 x_2' + \frac{u}{\lambda_2 - \lambda_1} \quad (150)$$

Let define two new variables as;

$$x_1 = -\lambda_1 (\lambda_2 - \lambda_1) x_1', \quad x_2 = \lambda_2 (\lambda_2 - \lambda_1) x_2' \quad (151)$$

Time derivative of the equation (151),

$$\dot{x}_1 = \lambda_1 x_1 + \lambda_1 u, \quad \dot{x}_2 = \lambda_2 x_2 + \lambda_2 u \quad (152)$$

The solution of these differential equations is given as;

$$x_1 = (x_1(0) + u) e^{\lambda_1 t} - u, \quad x_2 = (x_2(0) + u) e^{\lambda_2 t} - u \quad (153)$$

Here $x_1(0), x_2(0)$ are the initial values. Now get rid of the time element in equation (153). Initially, write $e^t = ((x_1 + u)/(x_1(0) + u))^{1/\lambda_1}$ and then subtract this into x_2 .

$$x_2 = -u + (x_2(0) + u) \left((x_1 + u)/(x_1(0) + u) \right)^{\lambda_2/\lambda_1} \quad (154)$$

If the control signal is +1, then the equation (154) is equal to;

$$x_2 = -1 + (x_2(0) + 1) \left((x_1 + 1)/(x_1(0) + 1) \right)^{\lambda_2/\lambda_1} \quad (155)$$

On the other hand, if the control signal is negative, then the equation (154) is;

$$x_2 = 1 + (x_2(0) - 1) \left((x_1 - 1)/(x_1(0) - 1) \right)^{\lambda_2/\lambda_1} \quad (156)$$

The switching curve is union of the equations (155) and (156).

$$x_2 = (x_1/|x_1|) \left[1 - (1 - |x_1|)^{\lambda_2/\lambda_1} \right] \quad (157)$$

This is the switching policy derived in terms of the bang-bang control approach for the inverted pendulum problem.

Time Optimal Control Simulation Results

The dynamics of the plant with the control policy defined by the equation (157) is shown in the figure below.

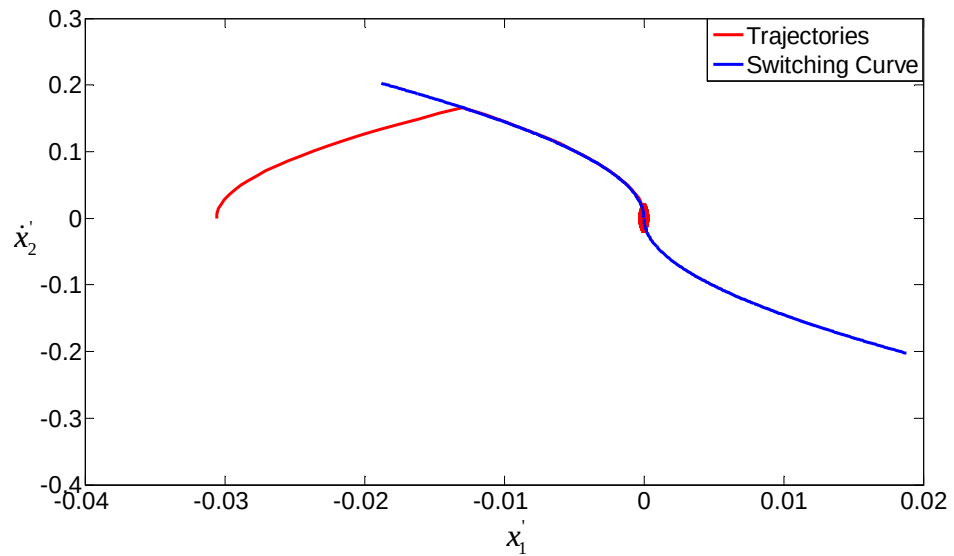


Figure 59: Dynamics of the inverted pendulum with continuous time optimal policy.

The initial state is $[-0.3 \ -0.3]$ and the length of the simulation is 100s. As described by the Figure 59, states evolve from the initial state to the switching curve and when they hit the switching curve, the control policy changes its sign. Therefore, the states reach the equilibrium point but due to persistent control signal they move in a bounded region. Now let see how the dynamics of the plant change if this control policy is applied in the discrete form. The figure below shows the dynamics of the discrete inverted pendulum with the control policy (157).

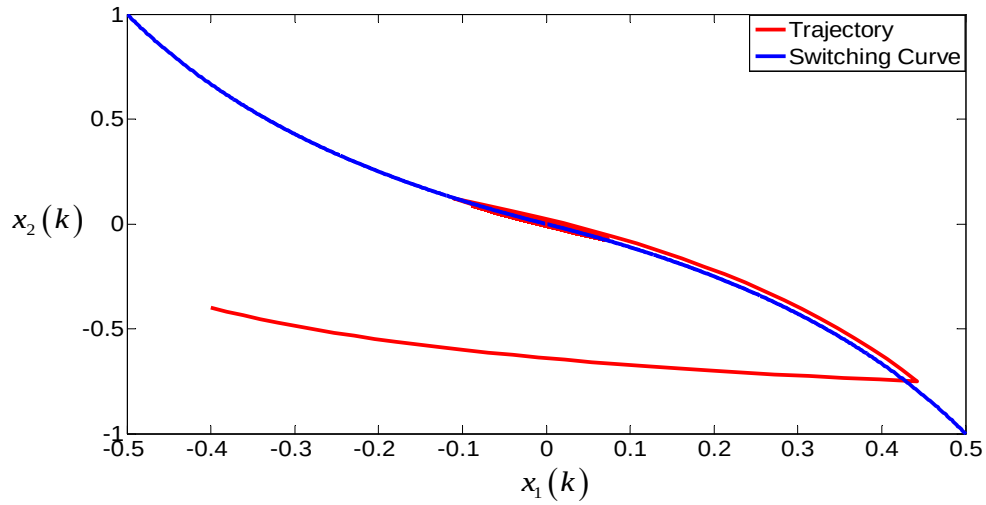


Figure 60: Trajectories of the inverted pendulum with the time optimal control.

The discrete dynamics of the pendulum with 0.01 sampling time is

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \end{pmatrix} = \begin{pmatrix} 1.0318 & 0 \\ 0 & 0.9691 \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} + \begin{pmatrix} 0.0318 \\ -0.308 \end{pmatrix} u_k \quad (158)$$

Initial states $\begin{bmatrix} x_1(0) & x_2(0) \end{bmatrix} = \begin{bmatrix} -0.4 & -0.4 \end{bmatrix}$, the length of the simulation is 20000.

As shown in the Figure 60, states of the inverted pendulum evolve from the initial state to switching curve. In this case control signal is +1, but when the states hit the switching curve control signal changes its sign and becomes -1. With this control signal, trajectories follow the switching curve and reach the target states. However, since the control signal is permanent, the states start banging around the target states, this causes occurrence of chaotic dynamics. The bounds of the region can be obtained from the figure below.

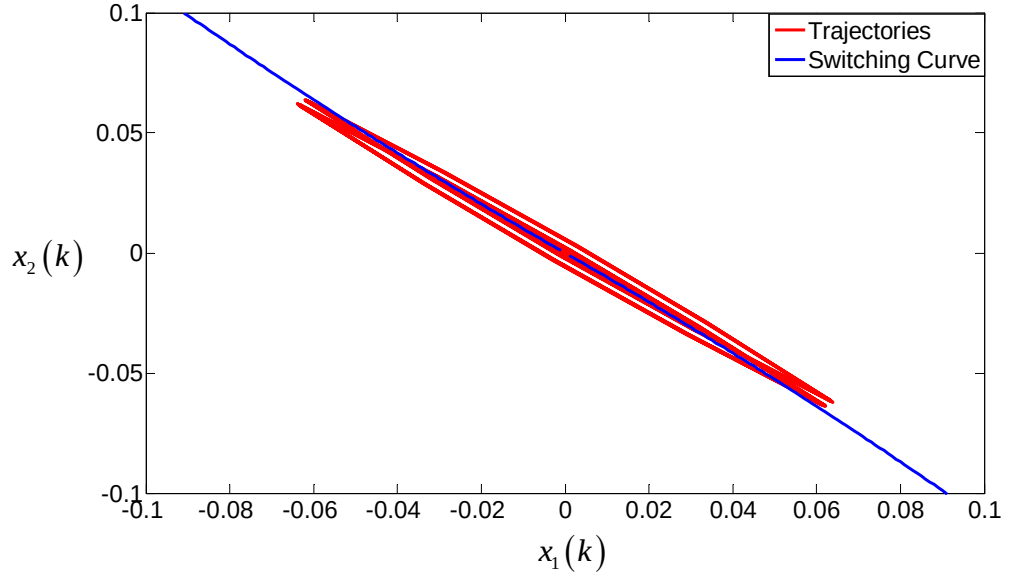


Figure 61: Bounds of the chaotic region for the inverted pendulum.

Initial state is $[0 \ 0]$ for the inverted pendulum and the length of the simulation is 1000. As can be seen from the Figure 61, the chaotic dynamics of the system is bounded around ± 0.06391 and evolve between 0.06391 and -0.06391. Again due to discrete time control effect, switching does not occur exactly on the switching curve.

Appendix D

The solutions for the joint position angles are obtained from the MapleSim MultiBody analysis template.

$$\begin{aligned}
theta1_sol := & \left\{ R1_theta(t) = \arctan \left(\frac{1}{Tx} \left(-Ty^2 - Tx^2 \right. \right. \right. \\
& + \frac{1}{Ty^2 + Tx^2} \left((Tx^2 Ty + Ty^3 \right. \\
& + \sqrt{-2 Tx^4 Ty^2 - Tx^2 Ty^4 + 4 Ty^2 Tx^2 - Tx^6 + 4 Tx^4}) Ty \left. \left. \right) \right), \\
& \frac{1}{Ty^2 + Tx^2} \left(Tx^2 Ty + Ty^3 \right. \\
& + \sqrt{-2 Tx^4 Ty^2 - Tx^2 Ty^4 + 4 Ty^2 Tx^2 - Tx^6 + 4 Tx^4} \left. \right) \left. \right), \\
R2_theta(t) = & \arctan \left(\frac{1}{2} \frac{Ty^3 + Tx^2 Ty}{Tx} \right. \\
& + \frac{1}{4} \frac{1}{Tx (Ty^2 + Tx^2)} \left((-2 Tx^2 - 2 Ty^2) (Tx^2 Ty + Ty^3 \right. \\
& + \sqrt{-2 Tx^4 Ty^2 - Tx^2 Ty^4 + 4 Ty^2 Tx^2 - Tx^6 + 4 Tx^4} \left. \right), \frac{1}{2} Ty^2 \\
& + \frac{1}{2} Tx^2 - 1 \left. \right) \left. \right\}, \left\{ R1_theta(t) = \arctan \left(\frac{1}{2} \frac{1}{Tx} \left(-Ty^2 - Tx^2 \right. \right. \right. \\
& - \frac{1}{Ty^2 + Tx^2} \left((-Tx^2 Ty - Ty^3 \right. \\
& + \sqrt{-2 Tx^4 Ty^2 - Tx^2 Ty^4 + 4 Ty^2 Tx^2 - Tx^6 + 4 Tx^4}) Ty \left. \left. \right) \right), \\
& - \frac{1}{2} \frac{1}{Ty^2 + Tx^2} \left(-Tx^2 Ty - Ty^3 \right. \\
& + \sqrt{-2 Tx^4 Ty^2 - Tx^2 Ty^4 + 4 Ty^2 Tx^2 - Tx^6 + 4 Tx^4} \left. \right) \left. \right), \\
R2_theta(t) = & \arctan \left(\frac{1}{2} \frac{Ty^3 + Tx^2 Ty}{Tx} \right. \\
& - \frac{1}{4} \frac{1}{Tx (Ty^2 + Tx^2)} \left((-2 Tx^2 - 2 Ty^2) (-Tx^2 Ty - Ty^3 \right. \\
& + \sqrt{-2 Tx^4 Ty^2 - Tx^2 Ty^4 + 4 Ty^2 Tx^2 - Tx^6 + 4 Tx^4} \left. \right), \frac{1}{2} Ty^2 \\
& + \frac{1}{2} Tx^2 - 1 \left. \right) \left. \right\}
\end{aligned}$$

Modelical code for the inverse kinematics model for the robot leg is:

```
model inverseKinematicsModel
```

```
  extends Maplesoft.Icons.CustomComponent3D;
```

```

Modelica.Blocks.Interfaces.RealInput Tx
  annotation(Placement(transformation(origin = {-100,50}, extent = {{-10, -10},{10,
10}},rotation = 0)));
Modelica.Blocks.Interfaces.RealInput Ty
  annotation(Placement(transformation(origin = {-100,-50}, extent = {{-10, -
10},{10, 10}},rotation = 0)));
Modelica.Blocks.Interfaces.RealOutput theta[4]
  annotation (Placement(transformation(origin = {100,0}, extent = {{-10, -10},{10,
10}},rotation = 0)));
protected
  Real th11,th12,th21,th22;
  equation
    th11=atan2((-Ty^2-Tx^2+(Tx^2*Ty+Ty^3+sqrt(-2*Tx^4*Ty^2-
Tx^2*Ty^4+4*Ty^2*Tx^2-Tx^6+4*Tx^4))*Ty/(Ty^2+Tx^2))/Tx,
(Tx^2*Ty+Ty^3+sqrt(-2*Tx^4*Ty^2-Tx^2*Ty^4+4*Ty^2*Tx^2-
Tx^6+4*Tx^4))/(Ty^2+Tx^2));
    th21=atan2((1/2)*(Ty^3+Tx^2*Ty)/Tx+(1/4)*(-2*Tx^2-
2*Ty^2)*(Tx^2*Ty+Ty^3+sqrt(-2*Tx^4*Ty^2-Tx^2*Ty^4+4*Ty^2*Tx^2-
Tx^6+4*Tx^4))/(Tx*(Ty^2+Tx^2)), (1/2)*Ty^2+(1/2)*Tx^2-1);
    th12=atan2((1/2)*(-Ty^2-Tx^2-(-Tx^2*Ty-Ty^3+sqrt(-2*Tx^4*Ty^2-
Tx^2*Ty^4+4*Ty^2*Tx^2-Tx^6+4*Tx^4))*Ty/(Ty^2+Tx^2))/Tx, -(1/2)*(-Tx^2*Ty-
Ty^3+sqrt(-2*Tx^4*Ty^2-Tx^2*Ty^4+4*Ty^2*Tx^2-Tx^6+4*Tx^4))/(Ty^2+Tx^2))
    th22=atan2((1/2)*(Ty^3+Tx^2*Ty)/Tx-(1/4)*(-2*Tx^2-2*Ty^2)*(-Tx^2*Ty-
Ty^3+sqrt(-2*Tx^4*Ty^2-Tx^2*Ty^4+4*Ty^2*Tx^2-
Tx^6+4*Tx^4))/(Tx*(Ty^2+Tx^2)), (1/2)*Ty^2+(1/2)*Tx^2-1);
    theta = {th11,th12,th21,th22};
end inverseKinematicsModel;

```